

;login:

November / December 1992 Vol. 17, No. 6

From the Editor's Desk

My friend Kevin C. (high school senior; science fair winner) was completing various scholarship applications today -- one wanted an essay on how his education would "best benefit society".

How does anyone "benefit society" and how would we know? Some benefactors are relatively easy to spot: Jesus, Buddha, Mohammed, Lincoln, Churchill. But how do we spot today's "hot contributors"?

Benefit appears difficult to measure. One way (of interest to ambitious students) concerns comparing annual monetary compensation. Ignoring contributors like Mother Theresa and Martin Luther King, it does raise an interesting question: How does one characterize the differences among high and low salary jobs?

We painted 'high salary' positions as corporate executives, doctors, and successful new-car sales people. Flipping burgers was the prototypical 'low salary' position for our discussion (though paleontologists were occasionally cited for their lack of high-paying jobs).

We discussed several properties: education, credentials, subordinate/superior relationships, and skills. The successful car sales person and the poorly-paid paleontologist contradict good correlations for all these factors.

My ex-boss Bill Wallace pointed out years ago that 'leverage' is one of the dominant factors determining compensation. When managers and other high-level people make a decision or policy, it impacts a larger part of society (in either human or financial terms) than decisions at the lowest level of burger flipping. Certain highly qualified scientists have important careers requiring skill, education, credentials, and dedication -- but are remunerated poorly for their lack of a larger impact on society.

I ask myself: "How can I increase my positive leverage in the big picture?" It's a tough question that appears to have important impacts. I'll let you know if I figure out any answers. RK

The closing date for submissions to the next issue of ;login: is December 16, 1992.

Association News

C++ Conference Report.....	3
<i>Susan E. Waggoner</i>	
Reader's Survey	5
President's Letter.....	6
<i>Stephen C. Johnson</i>	
Conference Meeting Space.....	7
Letter to the Editor.....	7

SAGE News

SAGE Report: LISA VI Conference	8
SAGE Views	10
SAGE Book Reviews:	
UNIX for Super-Users	13
Practical UNIX Security.....	13
SAGE Working Groups	15
SAGE Membership Order Form	19

Features

Privacy-Enhanced Electronic Mail.....	20
<i>Matt Bishop</i>	
A Comment on Open Systems.....	22
<i>Geoff Collyer</i>	
A Perspective on <i>pathalias</i>	23
<i>Rick Salz</i>	
Nostalgia: USENET Map	26
<i>Karen Shannon</i>	
Distributed Multi-Media.....	28
<i>George Neville-Neil</i>	
Modula-3	30
<i>Paul Vixie</i>	
An Update on UNIX-Related Standards Activities.....	35
<i>Stephen Walli</i>	

Book Reviews

The Bookworm	40
<i>Peter H. Salus</i>	
Crossing the Internet Threshold	
The Internet Companion.....	41
<i>Billy Barron</i>	

CD Offer	42
----------------	----

Announcements

SEDMS IV.....	43
Summer 1993 Conference	44
Mobile Computing Symposium	46
Winter 1993 Conference	47
Call for Special Issues.....	48
Publications Order Forms.....	50
USENIX Online Index	55
Local User Groups	56
Calendar of Events.....	59



The UNIX and Advanced Computing Systems
Professional & Technical Association

General Information

login: is the official newsletter of the USENIX Association.

login: (ISSN 1044-6397) Volume 17, Number 6 (November/December 1992) is published bimonthly by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710. \$24 of each member's annual dues is for an annual subscription to *login*. Subscriptions for non-members are \$50 per year. Second-class postage paid at Berkeley, CA and additional offices. POSTMASTER: Send address changes to *login*, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

Contributions Solicited

You are encouraged to contribute articles, book reviews, and announcements to *login*. Send them via email to login@usenix.org or through the postal system to the Association office. Send SAGE material to bigmac@erg.sri.com. The Association reserves the right to edit submitted material. Any reproduction of this newsletter in its entirety or in part requires the permission of the Association and the author(s).

Editorial Staff

Rob Kolstad, Editor <kolstad@usenix.org>
Ellie Young, Staff Editor <ellie@usenix.org>
Carolyn S. Carr, Managing Editor and Typesetter <carolyn@usenix.org>
Stephe Walli, Standards Report Editor <stephe@mks.com>
Bryan MacDonald, SAGE Editor <bigmac@erg.sri.com>
Michelle Dominijanni, Copy Editor

Membership and Publications

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Telephone: 510/528-8649
FAX: 510/548-5738
Email: <office@usenix.org>

Copyright © 1992 USENIX Association. The Association acknowledges all trade references made herein.

login: was produced on a SUN 3/50 with Framemaker 3.0 software and printed on recycled paper. ♻️
Thanks to mt Xinu for technical assistance and laser output from their Apple Laserwriter IIg.

Conferences, Workshops & Symposia

Judith F. DesHarnais, Conference Coordinator
USENIX Conference Office
22672 Lambert Street, Suite 613
El Toro, CA 92630
Telephone: 714/588-8649
FAX: 714/588-9706
<conference@usenix.org>

Tutorials

Daniel V. Klein, Tutorial Coordinator
Telephone: 412/421-2332
<dvk@usenix.org>

Supporting Members

Digital Equipment Corporation
Frame Technology, Inc.
Matsushita Graphic Communication Systems
mt Xinu
Network Computing Devices, Inc.
Open Software Foundation
Quality Micro Systems
Sun Microsystems, Inc.
Sybase, Inc.
UNIX System Laboratories, Inc.
UUNET Technologies, Inc.

USENIX Association Board of Directors

If a member wishes to communicate directly with the USENIX Board of Directors, he or she may do so by writing to board@usenix.org

President:

Stephen C. Johnson <scj@usenix.org>

Vice President:

Michael D. O'Dell <mo@usenix.org>

Secretary:

Evi Nemeth <evi@usenix.org>

Treasurer:

Rick Adams <rick@usenix.org>

Directors:

Eric Allman <eric@usenix.org>

Tom Christiansen <tchrist@usenix.org>

Lori Grob <grob@usenix.org>

Barry Shein <bzs@usenix.org>

USENIX Association

Executive Director

Ellie Young <ellie@usenix.org>

C++ Conference Report

by Susan E. Waggoner

<null!susan@sparky.imd.sterling.com>

The 1992 USENIX C++ conference was held in a steamy Portland, Oregon, on August 10-14. The exceptional weather was no deterrent to attendees and presenters who participated in a conference clearly aimed at successful production and support of C++ software. The majority of presentations concerned application support class implementations, library support strategies, and development tools. The business is heating up!

The conference was preceded by two days of tutorials covering object-oriented design, programming style and effective implementation techniques and reusability. The practical issues covered in these classes were extremely useful for new and experienced industry developers.

Keynote

The keynote address was given by Kristen Nygaard, the co-inventor of Simula and the new programming language Beta. The primary motivation for Simula came from the types of problems he was trying to solve in operations research. He needed a language that could describe dynamic systems with many components. The goal was to comprehend, describe, and communicate about systems; the result was a system for managing an interacting collection of nested objects, each with its own stack. In 1967 the idea of subclasses came from a problem with references to type.

Professor Nygaard was an active participant in the conference. His years of experience with developing object-oriented languages gave him great insight which he willingly shared.

Applications and Application support classes

Implementing efficient and effective classes for application support was a core issue at the conference. The presentations began with a paper by Daniel Edelson, "Smart pointers: They're smart, but they're not pointers," that discussed classes that simulate pointer types by using overloaded `->` and `*` operators. Such classes can be used as the basis for garbage collection systems and for persistent objects. The author then described the differences between the behavior of smart pointers and the built-in pointers of the language. The two main areas of difference are in supporting

pointers to const objects and standard pointer conversions in class hierarchies.

Building on the smart pointer mechanism was a garbage collection implementation. It was designed as a smart pointer template class. Although performance was admittedly a weakness, this implementation was compiler independent and was designed to be able to support different collection algorithms.

Another support class built on templates was a method for recursive iteration (e.g., depth-first traversal of a graph) where an iterator object must maintain state between visits to the nodes. This article details the analysis of the problem and also describes its solution via a language extension, discussing how a careful search for a solution within the existing language can obviate a proposed extension. The presentation was both an amusing and important discourse.

Communication between remote objects is an important problem for distributed applications. One session was based on three papers proposing different approaches for incorporating RPC/XDR functionality into C++ applications. One design was to integrate the protocols into the C++ I/O streams model, another was to have objects wanting to do remote access inherit the functionality from an abstract class.

Andrew Koenig, the chairman of the first C++ conference back in 1988, presented an elegant paper on a data structure for space-efficient representation of trees. Designed for storing C++ programs in a programming environment, the objectives were minimum space usage and quick traversal. The implementation eliminates pointer overhead and uses the adjusted sum of the sequence of the nodes to reconstitute the structure.

The problem of translating object-oriented data to or from a relational database led to the development of the O-R Gateway by Abdullah Alashqur and Craig Thompson. The gateway translates a relational schema to C++ classes (data members only) using mapping rules. Member functions must be added manually but the resulting classes can be used in C++ applications that need to access a relational database. The gateway also translates object queries into equivalent SQL queries, interacts with the relational database and translates the results into C++ objects.

An exquisite graphics video demonstrated the results of using C++ classes to develop shock-wave physics simulations. The developers at Sandia National Laboratories found that C++

abstraction and operator overloading facilities lent themselves to this set of problems. They found that once good base classes were developed they were very reusable. Performance approached FORTRAN and C implementations.

Library Support

Providing support for commercial libraries was the topic of two articles. The areas of concern were encapsulation techniques and linking of new versions without recompiling the application.

The InterViews class libraries developers have formulated various techniques for encapsulating their libraries. In order to be able to change the way an object is instantiated without affecting the user, a "kit" or creator class (sometimes called an object factory) is used. The library user calls a virtual member function of the kit class which then creates the object. This simplifies the interface and simplifies access. Other ways to avoid affecting user name space were also discussed.

Releasing new versions of dynamic libraries without requiring recompilation of the client applications was Andrew Palay's goal for Δ C++, developed at Silicon Graphics, Inc. This C++ system supports compatible class changes such as member-extension, class-extension, member promotion, and override-changing. It uses a vector of offset values to resolve references to class members.

Tools

Development environment and analysis tools are important for the continued success of C++.

SNIFF, developed by Walter Bischofberger at the Union Bank of Switzerland, is a portable C++ programming environment that provides browsing, documentation, and other support. It runs under several windows-based user interfaces and UNIX workstations. It is built using a fuzzy parser for information extraction and main memory stored symbol table for the information repository.

Alf (A Language Foundation) is an abstract representation for C++ programs. It was designed at AT&T Bell Labs to be the basis of a comprehensive programming environment. Design goals were static typing, abstraction, and a compact representation. It represents C++ program semantics as trees in Andrew Koenig's representation discussed above.

CCEL, the C++ Constraint Expression Language from Brown University, enables you to express design, implementations, and stylistic constraints on C++ programs that can't be expressed in the language itself. The Clean++ tool checks for violations of constraints in the program code. The CCEL implementation uses the REPRISE C++ semantics capture and representation system described at last year's USENIX C++ Conference.

Large C++ programs pay a price on startup for the initialization of static objects. John F. Reiser of Mentor Graphics has developed a system that reorders the functions in the program to localize the static initializers and can also be used to reduce page faulting by compactifying identifiable working sets of functions. As part of his talk he demonstrated a facility that graphically displays the pages touched during initialization.

Cdiff is a new tool in Judith E. Grass's CIA++ (C++ Information Abstractor) toolkit. It identifies significant syntactic difference between different versions of C++ programs. Its implementation depends on the database and definition queries of CIA++.

Extensions

The authors of μ C++ argue "It is not possible to build concurrency facilities from existing language features in C++ without sacrificing essential capabilities." Concurrency requires implementation at such low levels that type safety violations and integrity of the runtime environment is at risk. They give a complete overview of properties needed to implement concurrency and present their work on extending C++ to support concurrency.

The first implementation of an important new feature of C++, exception handling, was described in a paper by several authors from Hewlett Packard. The implementation has four functional areas: transfer of control, exception identification, object cleanup, and storage management. Control transfer from the point where the exception is thrown to the appropriate exception handler is done with the C library routines `setjmp()` and `longjmp()`. The correct handler is identified by matching the type of the exception object to the argument type of each candidate handler. `Typeinfo` objects containing base class information such as visibility and virtualness are allocated for all types where it is necessary to support this. Any automatic objects with destructors that are popped off the stack by the control

transfer must be properly destroyed. Destructor counters for objects and cleanup regions are used to guide the destruction process. Not surprisingly, run-time performance is better in a non-portable than a portable implementation of exception handling.

A complement to the paper describing the implementation of exceptions was a paper on an application of exceptions by Philippe Gautron of the University of Paris VI. His paper described a replacement for the venerable `assert` macro that causes an exception to be thrown when an assertion fails, thus enabling the application to attempt a recovery. A filter class was derived from the base class for testing. The filter class added the error handling functionality.

RTTI panel

The last event on the program was a panel discussion on run time type identification (RTTI), a controversial proposed extension to C++. Some developers believe that RTTI is necessary to well-designed object-oriented programs, as shown by the fact that it is "faked" in most major libraries. Others think that it will lead to bad designs, giving programmers a way to avoid doing it right when it takes too much thought. The panel members included Bjarne Stroustrup, Doug Lea, Jim Waldo, and Dmitry Lenkov with Mark Linton as chair.

Bjarne Stroustrup started the discussion with a presentation of the current status of the proposal. The most recent version of the proposal contains two parts: a run-time checked type operator (`?type-name`) and a `typeid()` operator that returns an object with run-time type information. The syntax of the checked type operator has taken a new direction since the last X3J16/WG21 stan-

dards committee meeting. The approach now may be to use template-like syntax:

```
check type-name(p)
```

and RTTI could be supported with a family of library templates.

Type information is stored in a `Type_info` class object. The minimal class definition should support comparison, return a name string, an ordering relationship, and an operator for accessing extended type information. The class could be extended by derivation to include information to support I/O, OODBMS, debuggers, etc.

Doug Lea presented a number of examples he developed to help illustrate the strengths and weaknesses of the RTTI proposal. The proposed extension will allow implementing dynamic argument-based dispatching and multi-methods. Heterogeneous collections will be more usable. Doug demonstrated how metaclasses could be simulated or logical states verified. RTTI cannot be used to infer features in classes that are not a part of the original design and it does not help support object persistence.

The third speaker, Jim Waldo, had some serious concerns about the extension proposal. He stated that RTTI is needed in three types of cases: object recreation, multi-methods or dynamic argument-based dispatching, and when the programmer is "fed up with objects" and wants to "violate the metaphor." He pointed out that the current proposal does not address the object re-creation problem, that dispatching on multiple argument types should be done with functionality more like virtual functions and you can already "violate the metaphor" with numerous programming techniques. He argued that it would be better to put off adding the functionality until alternatives are better understood.

Reader Survey

During the past year, *login:* has undergone changes in its editorial direction and format. We have an additional editor at the helm, Rob Kolstad, and a new typesetter, Carolyn Carr. We wish to seek out your opinions concerning the contents.

If you have the time, please review the last 5 issues of *login:*. Specifically, we would like your input regarding the following areas:

Do you find the articles in the "Features" columns to be poor, adequate, or excellent? How can we improve this section? What areas would you like to see more articles on? Would you like to contribute an article?

Please send your comments to: login@usenix.org. We look forward to hearing from you, so that we can continue to improve your newsletter in future issues.

Rob, Ellie, & Carolyn

President's Letter

by Stephen C. Johnson

<scj@usenix.org>

The prophets of doom and gloom are in the ascendency as I write this. Major computer companies are laying off thousands. The death of UNIX is predicted, or even announced, by everyone from *Byte* magazine to Microsoft. Windows NT is predicted to own 80% of the market in five years, even as it slips three months every three months.

I visited Microsoft last month, a most interesting visit indeed. It felt like visiting Sun in 1985. The people I spoke with were excited about the future, they felt that they were doing work that was advancing the computer industry, the world economy, and Human Knowledge. In short, they felt like winners. One person said to me: "I think a lot of companies are trying not to lose. At Microsoft, we are trying to win." As I said, it reminds me of Sun in 1985, or, for that matter, Bell Labs in 1975.

In the UNIX world, we have recently been trying to please everyone; at USL and OSF, many dozens of companies must agree on new features. We are paying the price of a 'Me decade' where everybody and his pet rat added incompatible 'improvements' to UNIX. The irony is that UNIX was made portable in the first place so that applications could move from machine to machine easily. Anyone who thinks we have achieved this should study the 4,000 line shell script which must be run to enable the Perl language to build in different environments. Perhaps we have fouled our nest beyond our ability to clean it.

In the midst of the gloom, several things seem clear to me. One is that this approach of 'playing not to lose' is going to fail. Companies that continue to do this will slide slowly, or sometimes

rapidly, into oblivion. There is no reason why large companies, or, for that matter, USL and OSF, can't articulate a vision and develop a strategy to realize that vision; when a group does this (as HP did recently) it begins to win.

Another thing that I believe is that at its heart our industry is still driven by technology. There are places for companies that deliver very low cost, or very high quality customer service, but at heart even these companies have to keep racing along with the technology or they will fail. We have had personal computers for less than a dozen years; within the last decade, the number of fax machines and cellular phones has exploded. More telling, of the companies that have done well in the recession, many have been at the leading edge of technology. Pen and Mobile computing, high speed wide area networks, multimedia, etc. will be upon us in no time.

I don't know whether the future will be Windows NT, UNIX, NextStep, or any of a dozen other contenders. I do believe that technology will play a major role in deciding the outcome. And I believe that USENIX will continue to communicate, teach, debate, and build the software technologies of the future, just as we have done with the technologies of today. The future will certainly mean change – technical change – but this is exciting as well as scary. So stop hedging your bets, quit your belly-aching, hop on a new technology, and shout your vision to the skies (or, at least, submit a paper to one of our conferences or workshops!). Whether you win or lose, that's the fun way to play the game.

USENIX Conference Meeting Space

By Ellie Young

There have been postings to *comp.org.usenix* pertaining to conference sleeping accommodations. Here is some information about how hotels are chosen and rates are set for USENIX conferences.

USENIX conferences place a heavy demand on hotel meeting space. During a typical week-long conference, we need many rooms to offer the 15-20 tutorials scheduled early in the week, as well as having two or three parallel tracks for several days following (some of which have over 1,000 attendees). The cost of renting this space in a major city would be astronomical.

What USENIX does is make an agreement with one or several hotels that, in exchange for a guaranteed minimum number of rooms rented by attendees, the hotel will throw in meeting space and other services at no direct cost to USENIX.

This agreement is complicated and, because the number of facilities in the U.S. where we can hold a meeting is rather limited, we typically sign these contracts five (5!) years in advance of a conference. These contracts contain penalty clauses (sometimes exceeding \$50,000) if we fail to fulfill our commitment to the hotel.

The hotel industry is fairly close knit, and word gets out quickly about groups that can't live up to their contracts. Such groups may have trouble getting space, or may have to pay a higher rate.

There are a couple of consequences of this arrangement. One is that it is important that when you make your reservation that you state that it is in conjunction with the USENIX Conference, and that you pay the USENIX rate, so we get credit towards fulfilling our room block commitment. (Even if you use a large corporation's discounted rate, it will not count as a credit towards the USENIX commitment.) Another consequence is that a widespread defection from our headquarters hotel will lead to higher registration fees for everybody.

Recognizing that not everyone can, or wants to, be in the headquarters hotel(s), we try to choose sites that have less expensive hotels in the neighborhood. We also encourage roomsharing (*comp.org.usenix.roomshare* is good for this purpose).

Letter to the Editor

by Rob Pike

<rob@research.att.com>

Dear Rob [Kolstad],

I was surprised that you disagreed on the headline [in the Sept. issue of *BYTE* magazine] reporting the death of UNIX.

Not only is UNIX dead, it's starting to smell bad. If not for the advanced state of decomposition, I'd recommend an autopsy. Even without one, though, I can predict what an autopsy would have revealed:

Cancerous pockets throughout; the entire system ravaged by malignant growths.

Opportunistic parasites in all major organs.

A grossly enlarged liver due to frequent systemic poisonings.

A failed immune system unable to respond to the onslaught.

Cause of death: a system utterly drained of vitality by the constant struggle of fighting the diseases.

One can only speculate on the mental health of the patient.

LISA VI Conference Report

by Steve Simmons

Inland Sea

<scs@lokkur.dexter.mi.us>

and Barbara Dyker

University of Colorado

<barb@locutus.cs.colorado.edu>

The Sixth USENIX Conference on Systems Administration (LISA) was held Oct. 21-23, 1992 in Long Beach, California. Approximately 600 people came from all over the world, including South Africa, Russia, and a fairly large contingent from Norway.

This year's conference dropped the official concentration on "large" installations. This was not due to lack of interest, but rather due to the increase in size of the average site. Sites which were considered startlingly large in 1988 are now the norm, and the attendees reflected this.

Doug Kingston's keynote addressed his experience at Morgan Stanley in replacing mainframes with a distributed TCP/IP network and UNIX workstations: over 500 systems spanning the world at seven key sites. The word it brought to mind is "sobering". Growth of their installation, as at most sites it seems, is expected to double in the very near term. Doug sees three key technological problems as limiting factors to growth: network bandwidth, disk i/o bandwidth, and memory speed. The technologies Doug sees as key in shaping the future are: window-based notification systems (email and wall don't cut it), automated electronic inventory control, caching network filesystems, faster networking (protocols, and controllers), and better system monitoring.

LISA seems to have reached a critical mass in both size and history. The papers and their presentation were uniformly good. The presenters were well-prepared, and audience at the individual presentations was attentive and had good questions. More significantly, LISA papers and presenters are building more and more on previous years work. Previous presenters brought a number of improvements to previous works, and a number of new presenters had modified and extended the work of earlier presenters.

Some notable items from the technical track:

"Effective Use of Local Workstation Disks in an NFS Network", presented by Paul Anderson, outlined a utility which caches server files on local disk based on how often each file has been accessed recently. The software is based on 'lfu'. In practice, he found a 70mb local disk cache can satisfy 90% of hits that would otherwise be nfs access to almost 1gb of tools on a remote server.

"LADDIS: A Multi-Vendor and Vendor-Neutral SPEC NFS Benchmark", presented by Andy Watson, has been in development for a considerable period and will soon be the indicator of preference for evaluating NFS server performance. LADDIS does not address NFS client performance.

"ipasswd - Proactive Password Security", presented by Jarkko Hietaniemi, introduced a new program to ensure good passwords. ipasswd is client/server based so the databases required for judging the password do not need to be replicated everywhere users change their password. This paper received the "Best Student Paper" award.

"Overhauling Rdist for the '90s", presented by Michael Cooper, provided insight into version 6 of rdist which provides considerable performance improvements and bug fixes.

The Wednesday afternoon session on "UNIX as the All-Purpose Computing Environment" had three excellent items. Peter Van Epp and Bill Baines of Simon Fraser University gave an interesting and sobering presentation on their successful effort at converting a 16,000-user site from a single mainframe with custom OS to a set of distributed UNIX systems. The paper was interesting in two different ways. One was sheer scope-- they had tight deadlines and a massive task which the apparently accomplished with flying colors. The other was the extent to which they brought mainframe-style attitudes, diagnosis, and analysis to bear on distributed systems and UNIX.

Van Epp and Baines were followed by a thoughtful paper by Peg Schafer on revisiting some of our traditional models for doing system administration. Schafer described BBN's success with using

a hierarchical model for administrators, with a dedicated central core of experts and lower-level or part-time specialists who serve given departments. The experiences with increased user satisfaction and localized control were encouraging.

Schafer's paper led very neatly into the following panel, "Models of System Administration". Schaefer, Rob Kolstad, Carol Kubicki, and keynote speaker Kingston were moderated by Pat Parsegian. Kingston seemed very much a proponent of centralized administration, while Schaefer represented the other end of the spectrum.

Elizabeth Zwicky offered another look at an old topic with "Typecast: Beyond Cloned Hosts." This work is in its relatively early phases, but it clearly already useful at her site. Zwicky presents a method and language for defining customization of machines such that more flexibility can be given to machine configuration at install time without losing the ability to quickly replace it with a clone.

Bryan Beecher has spent a great deal of time working on DNS and other name service for the University of Michigan Computing Center. He has developed various tools for dealing with incorrect DNS setups, and presented them and their use in a finely focused talk "Dealing with Lame Delegations".

Another useful tool set for the day-to-day administrator on the Internet is "Majordomo: How I Manage 17 Mailing Lists Without Answering -request Mail." Brent Chapman has centralized a number of functions for dealing with mailing lists into a single handler that lets different users manage different lists without requiring either root access or administrator intervention. Chapman received a number of suggestions both during the talk and afterwards at the Mailing List Managers workshop, and we will probably be seeing more of this in the future.

In addition to the technical track, there were two significant areas of general interest. For the second year there was the "Alternate Track," a series of mini-workshops on specialty topics. Attendance at these workshops ranged from 9 to 90.

Topics included managing mail and new parks, network administration, software installation issues, managing large mailing lists, etc. While there should be some format changes to improve the track, most workshop attendees thought they should be repeated in future years.

There were a number of evening functions which focused on the upcoming Board of Director elections for the System Administrators Guild (SAGE). On Wednesday evening the interim board presented a report on the history, purpose and current state of SAGE, including status reports from the individual working groups. Those reports can be found elsewhere in this issue. This was followed by a fairly lively question and answer period.

Nominations for the first elected Board of Directors closed Thursday afternoon. That evening Rob Kolstad moderated a debate/panel discussion with all the nominees who were at the conference. Each nominee gave a position statement; then, questions were taken from the audience.

Another new item for LISA was the Terminal Room. Barb Dyker and crew did yeoman duty in keeping things rolling in spite of balky phone lines and external routers. System administrators are especially sensitive to being "out of touch," and the access was much appreciated.

Chairman Trent Hein and his committee are to be congratulated. Each LISA has been better than the previous, and this one set a high standard for Bjorn Satdeva and next year's committee.

Best Paper Awards

The program committee at the LISA VI Conference selected the following paper as the best overall paper:

NFS Performance and Network Loading by Hal Stern and Brian Wong of Sun.

The best student paper award went to:

Jarkko Hietaniemi, Helsinki Univ. of Technology for *ipasswd - Proactive Password Security*.

Whither the Customer, Part Two

by Wendy Nather
Swiss Bank Corporation
Zurich, Switzerland
<wendy@sbcoc.com>

I would like to respond to the points made by Kevin Smallwood and Rob Kolstad in this column in the previous issue of *login*:. In general, I agree with both of them: users should be treated as customers, and the customer isn't always right.

As part of a systems administration team responsible for supporting an 1800-node options trading network in sixteen cities around the world, I have seen that a higher level of commitment to the customer does pay off. The willingness to (1) be available and (2) tackle a problem even if it's not your job is a valued trait that is not necessarily found in the corporate cultures in other countries.

I believe that more system administrators are making themselves more available than ever before. Witness the number of portable phones and beepers at the San Antonio USENIX conference. Sure, they're cool toys, but the mystique wears off quickly after a couple of 3:00 am calls from Tokyo. As UNIX spreads even further in the industry, more time-critical systems are using it, and we can no longer afford the luxury of maintaining research systems in a quiet corner of a university during spring break.

Nevertheless, the greater the UNIX presence, the more often we find ourselves maintaining systems for non-technical customers. There is perhaps no greater problem facing system administrators today than the fact that our managers don't know what we do. They don't appreciate the effort, the skills involved (how many managers have suggested that instead of hiring a new UNIX administrator, you just train the filing clerk who has a little extra time on his hands?), or the number of personnel needed to provide the best service possible. The average customer these days sees his computer as a pencil: he doesn't care why it's broken; he wants a new one, and he wants it NOW.

But I've found a simple phrase that often makes all the difference to a customer, no matter how lit-

tle he knows about the system, and no matter whether the problem falls within your jurisdiction or not: "I'll do my best." It does not promise the customer the moon and the stars; it does not even necessarily promise that you'll fix the problem within a certain time. It does convey your commitment to service, and it is an amazing contrast to the phrases I hear all around me in other cultures:

"He's in a meeting." "He's on vacation for three weeks. No, there's no one taking his place." "It's after five; he's gone home." "I don't know how to do that." "I can't do that." "That's not allowed." "It's impossible."

When you indicate that you're willing to tackle the problem WITH the customer, you create the teamwork that Rob Kolstad describes and defuse the power play and adversarial situations. Customers become immediately more flexible and tolerant when they realize that you really are on their side.

Note that this does not mean accepting rude or abusive treatment from a customer (user). My managers all the way up the ladder have come to the defense of the support groups and quietly insisted on consideration and respect. Both customers and system administrators deserve the same treatment.

Not only does the attitude "I'll do my best" improve life for the customer, but for support staff working together as well. My job would be intolerable without the customer orientation of the staff in the other support groups at my firm. When I ask them for help, I feel like a customer myself.

If you are a system administrator, then supporting the users of that system, whether directly or indirectly, is part of your job. How willing are you to do your job? The answer means the difference between a Mickey-Mouse operation and Disneyland.

The Customer Isn't Always Right; the Customer Isn't Always Even a Customer

by Elizabeth Zwicky
SRI International
<zwicky@erg.sri.com>

Kevin Smallwood makes some interesting arguments in the previous issue's column for calling the people who use the computers you are responsible for "customers" instead of "users." At my site, at least, the correct name for these people is "colleagues." My customer – the entity that pays me to administer computers – is SRI International. There's a very important distinction, there.

Kevin points out that if you go to Disneyland and drop an empty film box on the ground, a Disney employee will smilingly whisk it away, and correctly identifies this as an example of Disneyland's excellent customer service, part of what makes Disneyland such a popular place to go. If you go to Yosemite and drop an empty film box on the ground, and a park employee is standing nearby, do not expect the same smiling service; expect a ticket for littering. Yosemite and Disneyland have chosen different sets of priorities. Disneyland has chosen to take a specific set of people, charge them large amounts of money, and for this money provide them with a bright, clean, happy place. Yosemite has chosen to be open to as many people as possible, at as low a cost as possible, to provide them with the great outdoors with as little modification as possible, and to protect the wilderness for other future uses at the same time.

Most systems work more like Yosemite than like Disneyland. A public-access UNIX system has a Disneyland-style problem; the people who pay for the machine are the people who use it, and the goal is to make them happy. A corporate or educational UNIX system has a Yosemite-style problem; making the people who are using it happy is an important sub-goal, but may be secondary to other things (like cost) in the eyes of the machine's owners.

SRI, my customer, has chosen a set of priorities on the Yosemite side. Many of the things that I need to do in order to further SRI's interests do not particularly please the people who use SRI's computers. For instance, SRI believes that security is a

high enough priority that it should be allowed to override convenience; it believes that it is more important that disk space be allocated cost-effectively than that users never run into disk space limitations; it believes that Macintosh software should be bought for the entire division and not for individual users, and that this should be enforced by making packages legally available to everyone from a centralized space. Each of these decisions results in a certain amount of unpleasantness, which I am expected to subject other people to in order to please our joint employer. It's tough to think of someone as a customer when you're being paid to be mean to them as nicely as possible.

Furthermore, it's not productive. Treating these people as customers, instead of as colleagues, encourages them to ignore my areas of expertise. In this society, "the customer is always right." In fact, as we all know, the customer is often wrong. We deal with that partly by assigning new names to customers who can be expected to be wrong; if you buy medical assistance, you are a patient; if you buy teaching, you are a student; if you buy transportation, you are a passenger. These roles come with the expectation that you will defer to a doctor, a teacher, or a pilot, who will apply expertise that you do not have or choose not to exercise. The role of customer comes with the expectation that someone in the sales role will defer to you. It is not appropriate for the people who use the computers that I am responsible for to expect me to serve them, rather than advising and instructing them, when it comes to matters that involve those computers. Just as the passenger doesn't fly the plane – even if the passenger owns the plane – the user does not control the computer.

Fortunately for system administrators, managing computers involves considerably less risk to life and limb than flying airplanes. Unfortunately, this makes roles much less clear-cut. Calling people "users" encourages one extreme, where the computer belongs to the system administrator and everybody else serves as a source of stupid user stories. Calling people "customers" encourages the other extreme, where the computer belongs to the people who use it, and everybody else serves as a source of fascist administrator stories. In truth, the computer belongs to whoever bought it, and we're all in this mess together.

“What one quality do you value most in a System Administrator?”

by Paul Moriarty
cisco Systems, Inc.
<pmm@cisco.com>

[I was talking with people outside the main ballroom at LISA VI this week when I was asked about the best qualities of a Systems Administrator. I thought about this, and decided to pose it as a question for the newsletter, since it is a topic of interest to us all. Paul Moriarty submitted the following sessions. – SAGE Editor]

Since making the transition to the management side of systems administration, I have had the opportunity to interview many people as potential members of the Engineering Computer Services team at cisco Systems, Inc. In addition to the typical laundry list of technical skills, the two skills that I value most in a potential candidate are articulateness and a strong desire to work closely with the user community.

As systems administrators, our most visible interactions from the perspective of our customers (the user community) are either those where we must interface with them directly (i.e., solving their specific problem or answering a question) or those where a resource upon which they depend has suddenly become unavailable. The key to the success of an organization lies in how well the customers perceive that you interact in these situations.

The engineering user community at cisco comprises people with a wide variety of technical expertise, ranging from the extremely knowledgeable to those who only wish to use a computer to get their job done and couldn't care less about the underlying operating system as long as it doesn't get in their way. The successful systems administrator must understand these differences and be able to adapt his/her interactions in such a way as to neither offend the technical user by responding too simply nor overwhelm and baffle the novice with too much underlying detail. Responses that are clearly and effectively expressed will not only leave the user with an answer to their question, it will also make them feel that you truly understand them and what they are trying to do. This fosters a sense that you are a member of their team as opposed to simply an answering service of some sort.

Every user knows what they want their computing environment to do for them and it is important for us as systems administrators to ensure that they get the most productive environment

that we can provide. However, the challenge lies in the fact that the users often cannot express their desires in a way that is easy for us to understand. It is not up to them to figure out how to communicate this effectively to you; they have tasks and commitments that already fill their day. It is up to you to develop an understanding of how they use the computing environment and devise ways to maximize their use. This can only be accomplished by talking with the user community and providing them with a forum where they can explain to you just what it is they do and how they use computers to do their jobs. The successful systems administrator will proactively establish dialogues with his/her customers and not merely try to deduce what it is they do from fixing their problems when they occur.

Computers don't always work correctly and most users understand and accept this. However, when the machines are not working properly, it is imperative that we let them know that something is wrong and that we are trying to remedy it. It is comforting and reassuring to them that the problem did not magically go away (and will likely come back) – yet I have seen many instances where a systems administrator will identify and fix a problem but not tell anybody about it. This communication is especially important when the problem is transient in nature and difficult to troubleshoot. Update the user community regularly on what you are doing to fix the problem, even if it means telling them that you haven't made any significant progress. It is surprising to see just how understanding and patient they will be if they know that you haven't forgotten about it (and if you fail to update them regularly, I can assure you that this is exactly what they will assume).

For many organizations, the days when a service organization could exist solely on its service metrics are gone. To be a vital part of the organization, we must add value as well. From the organization's perspective, the only way that we as systems administrators add value is if our customers perceive us in that way. Thus, in order to be successful systems administrators we must not only be technically competent, we must understand our customer's needs and be able to articulately interact with them on their respective levels. The best way to accomplish this is to work closely with them, identifying their problems rather than acting as a background process, quietly fixing things or waiting for them to come to us with a problem or question before interacting with them.

SAGE Book Reviews

UNIX for Super-Users by Eric Foxley
(Addison Wesley, 1985, ISBN 0-201-14228)
pp. 213, \$32.95 softbound.

Reviewed by Elizabeth Zwicky
<zwicky@erg.sri.com>

I strongly recommend this book to anyone who wishes to administer a System V.2 or System III system in the 1980s. For those of us who are not time-travelers and do not maintain nearly-antique computer systems, this book is of little practical use. I wouldn't bother to review it if it weren't being sold, as a UNIX system administration book, in my local bookstore.

This is probably not going to confuse the experienced UNIX user, who will rapidly note some oddities. For instance, since this book was written, the UNIX world has moved past the point where it is reasonable to write a book that attempts to cover system administration on all UNIX variants in 210 pages, including an appendix listing all major commands, and one on the C preprocessor. Additional time has also added unintentional humor value to statements like "The BSD releases have always been very close to standard Bell distributions in respect of all conventions and standards." It would be nice to say that statements about the tendency of vendors to introduce incompatible changes and call them improvements were equally dated.

The rest of the book is a similar mix of eternal truth and painful anachronism. It's hard to seriously consider advice on backups from a book that still spells "restore" without the final "e". On the other hand, du, df, and quot have grown no new vowels in the last 7 years, nor are there significant new tools for managing disk space. No recent book would advise creating a "guest" account, even one that was not accessible through the network or modems, but the explicit security advice has held up pretty well.

A modern system administration book would also choose its topics somewhat differently. For instance, SCCS, make, and the C preprocessor, in their normal C development uses, are not likely to appear these days, when it is no longer assumed that one of the things that a system administrator does is maintain the operating sys-

tem source code. On the other hand, coverage of networks is considerably more extensive these days.

There is nothing fundamentally wrong with this book; for its time and its audience, it was a fine book. But it has gone the way of all technical books, and become a curiosity, rather than a reference. Do not be misled by its sparkling new appearance.

Practical UNIX Security by Simson Garfinkle and Gene Spafford, (O'Reilly and Associates, Inc., 1991, ISBN 0-937175-72-2) pp.481, softbound.

Reviewed by Dan Farmer
<zen@death.corp.sun.com>

Overview:

For many years there was only one book on UNIX security – Wood and Kochan's *UNIX System Security*. While quite informative and serviceable in its time, it became dated as UNIX technology (especially networking and network services) raced onwards. After the Morris Worm swept through the Internet, a new level of interest in security arose in the UNIX community, and within the last two years there have been several reasonably good books on UNIX security that attempt to close the information gap. *Practical UNIX Security* is such a book. It is not only a great security book, but it is a very good book, period – well written, informative, and entertaining – a real rarity in the computer publishing world.

Computer security awareness has blossomed quite a bit over the years; security books now not only go over the specific problems of a UNIX system, but they also delve into how to deal with legal problems, public relations, and the press when break-ins occur, what to do and who to contact when security problems arise, and a plethora of other issues. This book not only covers all this, but also goes over the basic fundamentals of UNIX security (the filesystem, networking) as well as several topics I'd never seen in print before (firewalls, discussions on specific weaknesses of various network services, etc.). The book gives frequent examples and code fragments (unfortunately not available via anonymous ftp, unlike most of the other code in books that O'Reilly publishes) to aid in the reader's understanding.

And while no book could possibly cover everything about UNIX security, *Practical UNIX Secu-*

ity covers all of the essentials. Indeed, a beginner might be daunted by the huge amount of information presented – certainly there are more than enough useful tidbits to occupy the mind of a more seasoned system programmer or administrator. The book attempts to solve this by including a fair bit of basic material in the beginning to get a neophyte past the low hurdles, but quickly moves onto more in-depth discussions of real security issues.

Typically, it starts each section by discussing the various problems with the topic at hand, then presents examples and sample programs to illustrate it in more detail, and finally goes into concrete reasons on why or why not the problem has a real solution or not. Refreshingly, the authors were unafraid to say the truth as they saw it – like, admitting that NIS is a security problem no matter what precautions you take. Too often security books will gloss over the protocols and merely echo what the vendors say about dealing with any potential security problems.

Content:

The basics are covered in the first few chapters – the filesystem, passwords, and various user and account problems. This is the most predictable/expendable part of the book, and nothing new is covered here. However, it does start giving some concrete C and shell source that provide some interesting insights on some of the more common problems you'll encounter on a UNIX system. It then moves into more interesting territory, that of actually securing your system against attacks and finding out if a potential break-in has occurred. This includes summaries of the ever important syslog mechanism, key log files, and still more source code. Among some tasty examples, there is a particularly clever script that tests to see if your system is susceptible to find and xargs attack (two commands that are typically run from cron). It's refreshing to find a book that expects the reader to use some thought to understand some of the concepts contained within, but doesn't expect you to do all the work – e.g., it never has the sometimes painful "this is left as an exercise for the reader" mentality.

The section on networks and communication is one of the key sections of the book. It covers modems, UUCP, NIS and NFS, Kerberos and Firewalls. Some of the highlights here include:

- The best section on discussing UUCP security I've seen anywhere, covering both Version 2 and HoneyDanBer UUCP.
- A complete chapter on the hows, whys, and

whats of "firewalling" (a technique that is used to isolate and secure your network from a hostile external environment by having only a small, tightly defended point of egress).

- A comparison of Kerberos and Secure RPC.

Details on how attackers can spoof and attack network services, with some tips on how to increase your security (and how sometimes that is impossible, due to the design of the service), along with a description of a few classic bugs (e.g., the wizard command in sendmail, fingered buffer overflow, etc.), are discussed here.

Finally, the book discusses how to discover a break-in and how to handle various intruder and programmed attacks, including monitoring and facing the intruders, keeping them out, and dealing with legal and law enforcement agencies should the need arise. Hints on how to find and contact other system administrators that might also be involved in the intrusion are provided.

Sections on cryptography (including some good descriptions of how DES and RSA work) and physical security (touting the all-important backup) are included. Several appendices contain valuable summaries, including an overview of how the insides of Kerberos work, checklists that include all the main points covered in each of the chapters, as well as important files (and their purpose) and processes on a typical UNIX system. An extensive bibliography and resource section completes the book.

Problems:

No book is flawless. This one tends to ignore most commercial software and hardware add-ons and solutions to various security problems, some of which are quite useful. Curiously, there was scant coverage of public domain software (other than obligatory references to Kerberos and COPS). In addition, security flaws or bugs (such as described in the CERT advisories) are given only a cursory examination. With the sophistication of attackers continually on the rise, it might have been wise to discuss and explore these problems in more depth.

Since the book is so thorough in other areas, some readers might assume that most of the problems here have been covered. While discussing bugs in any depth is a touchy subject, giving a brief description of just a few and not talking about the severe problems in this area seems almost more dangerous than not talking about them at all to me. However, I thought perhaps the most serious

flaw was a seeming lack of care about the importance of computer security policies -- a chapter or an appendix containing more information.

Only a few technical errors crept their way into the text (although the first edition had all the backquotes in the example shell scripts accidentally transposed to forward quotes). Since vendors offer continuously changing services (e.g., NIS+ by Sun Microsystems), UNIX security is a moving target that no book can hope to keep up with. It will be interesting to see if further edi-

tions of this book will attempt to keep up with the constant flow of new technology.

Summary:

The problems in the book are far overshadowed by its tremendous strengths. Well thought out, well-written, authoritative, opinionated, and fairly comprehensive, this is an excellent book that no system administrator or person interested in UNIX security should be without. Beginners might be a bit overwhelmed by all the material, but a bit of perseverance will be rewarding.

SAGE Working Groups

The following is a list of the status and progress by some of the various groups established within SAGE to help reach our members and define our future. If you are interested in hearing more about these groups, or making your voice heard in their directions and goals, join the mailing lists and talk with the leader of each group. You can send email to majordomo@usenix.org and ask for help in the body of the message to get further information on joining the working group mailing lists.

Status of SAGE mailing lists

The site usenix.org supports 17 SAGE-related electronic mailing lists: one main "sage" list and one for each of the 16 SAGE working groups (all named "sage-<groupname>", e.g., "sage-ethics"). The most popular lists are "sage" (132 members as of 11 October 1992), "sage-security" (50 members), "sage-jobs" (49 members), and "sage-policies" (46 members). To date, the lists have processed over 700 KB of traffic. Archives are kept, though they aren't currently available for anonymous FTP access; the SAGE-Online group is working towards a solution for that issue. At this time, participation on the mailing lists is not restricted to SAGE members; anyone can currently join any list.

The "Majordomo" automated mailserver manages all the mailing lists. (see Brent Chapman's paper on Majordomo in the LISA VI proceedings, or get it by anonymous FTP from [FTP.GreatCircle.COM](ftp://ftp.greatcircle.com/pub/majordomo.paper.ps.Z), in file "pub/majordomo.paper.ps.Z"). Majordomo is much like a BITNET "LISTSERV" server; it responds to emailed commands in the body of a message (not the "Subject:" line). To subscribe to a SAGE mailing list, send the command "subscribe <list> [<optional email address>]" (for example, "subscribe sage" or "subscribe sage-policies You@There.ORG") to the

address "Majordomo@USENIX.ORG". To find out more about what Majordomo can do, send the command "help" to "Majordomo@USENIX.ORG".

Brent Chapman, SAGE Postmaster

Conferences and Workshops

The Sage-Conferences working group includes twenty people-- over 80% of whom are commercially affiliated. A general statement of purpose was sent to the list with the preface:

The working groups were given a specific charter at the mass meeting in San Antonio. Each group is to focus on a specific topic area and attempt to answer one or more of the following questions: Is this in general an area that SAGE should be involved in? and if so, can a coherent statement of purpose be developed for that topic? Can the group make some specific recommendations about what SAGE should or should not be doing in this area? In all of the above, we should present a rationale for each decision. The rationale should include (where relevant) the reasons for and reasons against. This group will focus on SAGE activities which relate to conferences and workshops. It should cover those which are done by USENIX, those which are done by SAGE, and those done by outside organizations.

After some initial activity, things quieted down. I have not pushed discussions any harder, as there are some issues outside the group's control which either cannot be settled by the group or which are obviously premature. A shortened version of the discussion is included below, with the response from the group.

LISA Conference

The status of LISA was negotiated with the USENIX board as part of the establishment of

SAGE. USENIX retains "ownership" of LISA, is responsible for the running of the conference, and gets the profits (or losses). Starting with the 1994 conference, SAGE will help pick the chairman. The 1993 chair is Bjorn Satdeva, who will no doubt welcome working with SAGE.

There was general agreement with this. Several members did say it might be nice if SAGE someday took full control of LISA, but nobody suggested actually trying to change the current agreement. The members feel that LISA is pretty much their conference for system administration and want to work hard to keep it that way.

WCSAS

The World Conference on System Administration and Security is a new conference sponsored by FedUNIX. They held the first WCSAS in Washington DC earlier this year. Afterwards they discussed with USENIX and the ad-hoc SAGE board the possibility of cooperating at future WCSAS. There is no final resolution to this, but it looks promising. The negotiations are being handled by a group with real authority to do so (sign contracts, that sort of thing), so we're kind of on the sidelines.

My comment that these were the only relevant conferences known to SAGE generated a reply from Mark McGloughlin <mcglough@glas.rtsq.mot.com>. He pointed out correctly that currently nearly all the activity seems to be in the USA. Given the expense of intercontinental travel, joint conferences are unlikely in the near future. We should pursue some sort of activities in Europe and be ready to do the same for the Far East and other areas as levels of interest increase.

Networking Workshop

SAGE has submitted a proposal to the USENIX board for a networking conference. This is a proposal, not a decision to do anything. One working title was "Administering Networks on the Fringe," e.g., FDDI and ADM and T3, or simply sheer size. This will almost surely be a SAGE workshop.

This proposal has now moved forward, and at least one Sage-Confs member (Brent Chapman) has applied for chairmanship. The members were in favor of the conference.

My request for other action items generated several suggestions:

Putting the L back in LISA. Several people independently proposed redefining 'large' for 1993 or 1994 and attempting to either do a small separate workshop or a focus track at LISA. This last suggestion has already been mentioned to Bjorn Satdeva, chairman of 1993 LISA, and he was generally positive.

We should watch for relevant conferences which have some current or potential overlap with SAGE, and volunteer to help provide relevant conference items under an explicit SAGE imprint. This is already under way semi-formally with BOFs at the recent USENIX Security Symposium.

We have been informed that small workshops rarely make money. It was suggested that we try having multiple workshops at the same site on consecutive days. This might cut the overall costs enough to make more workshops feasible – effectively putting on two workshops for the price of one (or one-and-a-half). If the workshops had related topics, so much the better.

In general, the group has been cautiously enthusiastic. I expect that we will continue in the same vein.

*Steve Simmons, Chair,
SAGE Conferences Working Group*

Online

SAGE-Online has started by discussing possible technology to distribute high signal/noise (i.e., useful) information. WAIS, Gopher, Anonymous FTP, Mail Servers, and a few other protocols were mentioned. The conclusion was that there are enough tools in existence right now that it does not make much sense in the short-term to create our own set of protocols. Instead, we are leveraging off existing tools/protocols for the mechanics of information dispersal. For instance, Apple has volunteered to host a SAGE WAIS server, and Erik Fair is in the process of building that machine currently.

There has been much less discussion about what sort of information would be useful. The first information that was identified as useful was full text of the papers presented at LISA, and we have started the process of getting the relevant permissions and data; permission forms were included in the author packets for LISA VI, and have been sent to all presenters for LISA IV and V, with the

e-mail messages for LISA I-III expected by the time you read this. I would like to encourage people to think about what other sorts of information would be useful and to send suggestions to sage-online@usenix.org.

I have some long-term dreams for a vendor neutral bugs database, but this is something that has a number of technical and legal difficulties which will need to be overcome. I think we could wisely spend our time working on projects that we know we can do, and that will be useful right now.

Mark Verber, Chair, Online

Education

As of October 12, the SAGE-EDU mailing list contained 27 addresses (at least one of which is itself a list). Traffic has been fairly light thus far, and I'm hoping that the chance to meet face-to-face at LISA (for those who attend) will boost the group's energy.

So far, we've begun a sysadmin booklist (thus duplicating several other efforts – it's my hope we can focus more on the suitability of these books for instructional purposes than general-purpose references) and a list of commercial tutorial offerings (here the hope is to eventually have comments from attendees available – I don't know how well this will work, as it may be a sensitive issue). We've also started collecting descriptions of sysadmin courses and curricula currently being offered. Eventually these compendia will be available electronically on the SAGE server.

Pat Wilson, Chair, Education

Certification

There have been three simplistic models proposed for certification to date. None has really been fleshed out to any extent at all beyond some quick discussion. Any approach taken will require a long time to design and develop prior to presenting a recommendation to the SAGE board.

These models included:

- The "all or nothing" model. You're either certified or you're not. Viewed as difficult to do given the diversity of things that different sysadmins do at various sites.
- The "merit badge" model. Test for competency in specific areas (e.g., backups, account maintenance, UUCP, etc.) and recognize competency these small, specific areas. Competency in a number of areas would qualify one to be a Certified Sysadmin, or Certified Sr. Sysadmin, etc.

Viewed as easier to certify people but may lead outsiders to view system administration as a set of tasks instead of a career.

- The "guild" model. Establish an apprenticeship program and establish criteria for "passage" to journeyman and master levels. While "fuzzy" enough to maybe cover more of the certification issues (apprentice sysadmins in the program would have a mentor or number of mentors who would decide when competency has been achieved), it may be viewed by outsiders as "not serious enough."

Lately, there has been some discussion saying that certification is a bad thing and should be avoided. I think at this early stage, it is too premature to decide as it is easy to point out flaws in something that doesn't really exist.

It is my intent to keep the skeptics in the group, encouraging their participation to the fullest extent possible so that any future process presented to the board will have had input from both sides.

Paul Moriarty, Chair, Certification

Job Descriptions

Currently there are 48 members of the SAGE Jobs Working Group. This group has been evaluating SAGE's role in assisting system administrators with definitions of job descriptions. The working group believes that this is an area that SAGE should pursue. The focus of the group is to create multiple system administration job description suites that can be used as templates for those who are writing position descriptions for hiring purposes at their own site.

We have determined that the format for the job descriptions template will include core job description templates and check off skills. The core templates will cover generic experience and skills necessary for a given level position. The check off list will cover the additional skills that a given administrator may specialize in or that a site might require.

To date, the working group has made progress on both the templates and the check off list. The templates that are being defined are for novice, junior, intermediate, and senior system administrators. The check off list includes additional expertise in areas such as security, networking, programming, and different flavors of UNIX.

Tina Darmohray

Awards

Awarding a "best paper" at LISA VI was discussed, and the general consensus was that since the program committee has plans to do this, SAGE needn't get involved. It was agreed that the SAGE awards should have a scope larger than a single USENIX conference, over a yearly time frame. The mechanism for nominating and judging papers hasn't been decided and the discussions have centered around how large a task this could be.

The original intent of the group was to somehow acknowledge individuals who have made a contribution to the system administration profession. I think singling out exceptional papers is a good idea, as is providing a "reading list" for those who can't wade through all the conference proceedings. There are other ways of making contributions to the profession, and somehow these should be acknowledged as well. This topic hasn't been discussed to any degree on the list, and I will be bringing it up again. *John Detke*

Public Relations

This group has ten members, more than half of whom are SAGE or USENIX board members. The public relations working group is an evolutionary vestige, like the whale's hand. It corresponds to the public relations statement in the SAGE charter, which evolved from an earlier and much more widely defined public policy statement. During the course of that evolution, most of the areas of activity which would have distinguished it from those of the publication group disappeared. It is therefore our recommendation to fold this working group into the publications group. *Paul Evans*

Local Groups

The most important issues are the definition of the relationships between the local group and SAGE/USENIX, and what influence this should have on the SAGE Bylaws and the general Special Technical Group rules.

Another topic surrounds the kind of support the local groups require from SAGE. One kind might be insurance, which can be very expensive. It can be difficult to find meeting facilities without an insurance policy.

Membership fees were also a topic of discussion. If membership in the local group requires membership in SAGE, and membership in SAGE requires membership in USENIX, it can be expensive to join a local SAGE group.

Because there are also people who want to start local groups soon, I will be acting as a contact point for this. Anybody interested in starting a local group similar to Bay-LISA or BackBay-LISA can contact me, *bjorn@sysadmin.com*.
Bjorn Satdeva

Policies

So far, this group has sent out a call for submissions of existing policy documents. We've established a warehouse for the submissions, and Ken Nawyn has printed the first index of what's out there. There will be a second index shortly. These documents are being used as the foundation for a booklet of sample policies, to be available electronically and in printed form. The group is still discussing whether it is more practical to divide up suggested policies by the type of site, or by the type or resource.

I've also contacted a policies-security mailing list that was formed after the USENIX security symposium and let them know of our and sage-security's existence, as well as inviting them to join us.

The group has had several lively discussions of policy issues (the difference between formal and informal policies, the difference between procedures and policies, whether a policy that cannot be turned into a procedure is worthwhile).

Lee Damon

Vendors

This group has 22 members. Discussions have been severely hampered by the unwillingness of anyone other than the chair and SAGE board members to express any opinions at all, and have now been further stymied by the chair's leaving system administration and resigning his position.
Elizabeth Zwicky

Non-UNIX

The sage-outreach mailing list has been relatively inactive, and has very few members. This may be due in part to the target audience not knowing about SAGE, the mailing list, or being a part of the usual communication channels, i.e., the Internet. It is important that we establish some means of reaching this audience. We did get new members after the announcement in SysAdmin, and so I think trade publications, particularly those geared towards "other" systems, are a good potential. I'll be bringing up these ideas again on the list, and hopefully our new members will have some ideas along these lines. It is important for SAGE and USENIX to continue to reach out beyond the traditional UNIX community.

John Detke

SAGE Membership Order Form

SAGE Information and Application Form

USENIX recently launched its first Special Technical Group. The Systems Administrators' Guild (SAGE) is devoted to the advancement of systems administration as a profession. SAGE will recruit talented individuals to the profession, develop guidelines for the education of members of the profession, establish standards of professional excellence and provide recognition for those who attain them, and promote work that advances the state of the art and propagates knowledge of good practice in the profession.

USENIX and SAGE will work jointly to publish technical information and sponsor conferences, workshops, tutorials and local groups in the systems administration field. An interim board has been appointed and elections will be held after this LISA Conference to choose a new board, which will take office in January 1993. If you wish to join SAGE, please return this form to the address below.

Yes, I would like to join the USENIX special technical group SAGE, the Systems Administrators' Guild, as follows:

- I am a current USENIX member. I wish to join SAGE. Enclosed is \$25 to cover dues for one year. My membership number is _____.
- I wish to renew USENIX and join SAGE. Enclosed is \$90. \$65 for a one year individual membership in USENIX; \$25 for a one year SAGE membership).

Name _____

Address _____

City _____ State _____ Zip _____ Country _____

Phone _____ email address: _____

PAYMENT OPTIONS

Check enclosed payable to USENIX Association or SAGE.

Please charge my: Visa MasterCard  

Account # _____ Exp. Date _____

Signature _____

Outside the U.S.A.? Please make your payment in U.S. currency by one of the following:

- * Charge (Visa, MasterCard, or foreign equivalent)
- * International postal money order
- * Check - issued by a local branch of a U.S. Bank

USENIX Mailing List

- I do not want my address made available to other members.
- I do not want my address made available for commercial mailings.

Please mail this form to: USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
FAX 510/548-5738

Privacy-Enhanced Electronic Mail

by Matt Bishop

<Matt.Bishop@dartmouth.edu>

Introduction

This article describes how privacy-enhanced electronic mail (PEM) works, and how it provides authentication, integrity, and (when desired) confidentiality for electronic mail. As with most networking protocols, PEM bases its security on cryptography, because the physical security of most networks over which it is to be used cannot be guaranteed.

A Quick Tour of Cryptography

Cryptography can provide two services for the transmission of messages:

- confidentiality: the contents of the message are kept secret; and
- authentication: the identity of the sender is (or is not) as claimed.

Public key cryptography can easily provide both these services. Each user has a private key, known only to her, and a public key, known to everyone. Each pair of these keys has the property that applying one undoes the other. For example, with an "exponentiation cipher," the public key e and private key d have the property that: ... these look like this: ... $e \dots C = M \text{ mod } n$...

$$C = M^e \text{ mod } n \text{ and } M = C^d \text{ mod } n$$

where C and M are the ciphertext and message, respectively.

Now suppose Matt and Holly wish to exchange secret messages. Matt's public and private keys are e and d while Holly's are e' and d' , respectively. (In what follows, all exponentiation operations are done (mod n .) Matt takes his message M , raises it to the e -th power, and sends the result C to Holly. When Holly gets C , she raises it to the d -th power, and gets the original M back. Without knowing d' , it is "computationally infeasible" to determine the M corresponding to C . So Matt and Holly can be confident no one read the message in transit, as only Holly knows d' .

Authentication relies on the user's private key being kept private. If Matt wants to send Holly an authenticated message (so she knows it came from him), he takes his message M , raises it to the

d -th power, and sends the result to her. When Holly gets the result, she can raise it to the e -th power to get the original message. As it was encrypted using a key known only to Matt, she can be sure it came from him. (Note that anyone can do this, not just Holly.)

Not all public key cryptosystems provide both these abilities; many provide one or the other. The best-known public key cryptosystem, the RSA (or MIT) algorithm, does provide both, and hence the initial versions of PEM will use that cryptosystem. (It is an exponentiation cipher, and works like the one described above.)

The authentication step also provides integrity checking, since if any change is made to the message in transit, it will come out garbled after Holly has decrypted it. But given today's technology, encrypting a lengthy message using RSA is too time-consuming, so messages are usually run through a one-way cryptographic hash function. This function generates a short (~64 to 512 bits) output; further, given a message M and a hash H , it is "computationally infeasible" to find another message M' that produces the same hash H . Then the hash is encrypted using the sender's private key, and is sent along with the message. The recipient simply recomputes the hash, then decrypts the transmitted hash using the sender's public key. If they match, the message was not altered.

What is Privacy-Enhanced Electronic Mail?

The term "privacy-enhanced electronic mail" (or "PEM") refers to the set of security services with which (the successors to) RFCs 1113, 1114, and 1115 augment SMTP. They use public-key cryptography, and assume recipients' and senders' public keys are available. (More on that later.)

The three basic services provided are:

- authenticity: that the sender or senders are really who she or they claim to be;
- integrity checking: that the message has not been altered since the sender or senders sent it and
- confidentiality: that the contents of the message cannot be determined while the message is in transit.

The first two services are always provided; the third is optional.

To send mail using these services, a user enters the message normally, and then requests that the security services be provided to it. Call the body (NOT the headers) the "message body." This message body is replaced by two parts: the PEM headers and the PEM body. Among other things, the PEM headers specify what public keys (the senders' and, where appropriate, the recipients') are being used. The PEM program puts the message body into a canonical form, and then generates a cryptographic hash that is then encrypted using the recipient's private key. This hash is then placed in the message as a field in the PEM header. If confidentiality is required, the message body is encrypted using the DES, and the DES key (the "session key") is encrypted using the recipient's public key and is placed into another PEM header field. The canonical (or encrypted) form is then transformed to printable characters and sent as the PEM body.

When a recipient receives a message, the PEM software validates it as follows:

1. Determine how the cryptographic hash was computed, and the sender's public key.
2. If the message is encrypted, decrypt the session key using the recipient's private key.
3. Decrypt the transmitted cryptographic hash using the sender's public key.
4. Compute the cryptographic hash expected using the PLAINTEXT (unencrypted) message body.
5. Compare the computed and transmitted hash values. If they match, the sender is the person whose public key was used to decrypt the transmitted hash (presumably, the same as named in the headers!) and the message was not altered in transit. Otherwise, there is a problem somewhere.

Compatibility Issues, Part I

Once a PEM body has been printably encoded (the last step of converting the message body), people not using PEM cannot read the message easily. If the message was encrypted, this is fine, as only the recipient for whom it was intended (and who, presumably, has PEM) should be able to read it. But if only authenticity and integrity are required, recipients without PEM should be able to read the message (although, of course, they will be able to verify neither its authenticity nor its integrity).

If some recipients do not have PEM, the sender can skip the printable encoding step during the

conversion of the message body into PEM headers and a PEM body, so the PEM body is simply the message body. But this may pose problems for recipients with PEM who try to verify authenticity and integrity, because some implementations of SMTP will add blank lines to the message, or change tab characters to blanks, or otherwise transform the contents of the message. These transformations do not alter the meaning of the contents of the message, but will cause the cryptographic hash function to produce a different value for the received message than it produced for the message when it was sent (which is proper, as the message has been altered in transit). So if all the recipients have PEM, it is more reliable to do the printable encoding.

Mailing Lists and Authentication

PEM can be used to send authenticated, integrity checked messages to all members of a mailing list. Most often, the sender will want the message to be authenticated and integrity-checked as coming from her. If so, the exploder (which forwards the letter to the members of the mailing list) need do nothing; any recipient can simply get the sender's public key and verify the cryptographic hash using it.

In addition to the above, the sender may want the message to be authenticated as having been sent to the exploder. For this, the exploder itself will need a public and a private key. The exploder must enclose the received message in another PEM message. Then the recipients first authenticate that the message came from the exploder, and extract the body; they then authenticate the original message as in the first case. (Nested PEM messages are allowed under the successor to RFC1113, and conforming implementations must be able to handle them.)

Authentication of requests sent to a list server is of course trivial; just use the normal PEM authentication mechanism.

Mailing Lists and Confidentiality

Sending a confidential letter to members of a mailing list is more difficult, because the session key used to encrypt the message must itself be encrypted using the recipients' key – and as (presumably) the sender does not know the full list of recipients when she composes the letter and sends it to the exploder, she cannot do so. This leaves two possibilities.

The first would be to give the exploder itself a private and public key. Then the sender merely sends a confidential message the exploder. The

exploder decrypts the session key using its private key, and re-encrypts it using the public key of each recipient; this would require two extra PEM header lines per recipient. This seems simple enough, but it raises a knotty problem. The protocol says that a message can have at most one originating entity. Now, in the above, which is the originating entity – the sender or the exploder? And if it is the exploder, how can the recipients know who the sender is? Finally, if there are two different originating entities, which one came first? So the obvious scheme fails.

The second is to have the exploder extract the session key, and decrypt the message. Then it encapsulates the message, unencrypted, into a PEM body, and applies confidentiality to that body in the usual way. This produces a nested message, which PEM implementations must be able to handle. The one problem with this scheme is the message will lie unencrypted on a host that (most likely) neither the originator nor the recipient (or their respective organizations) have control over.

So they most likely have no idea of the quality or type of the security services offered by the operating system. Hence it requires some degree of trust in the system on which the exploder resides.

Sending a confidential letter to the list server can be done using ordinary PEM features.

Conclusion

This article presents a quick overview of how PEM works. Future ones will describe the specifics of the PEM headers and how public keys are bound to an entity.

Given the increased reliance being placed on electronic mail, and its complete lack of security, people will become more and more concerned about the dangers of using unsecured electronic mail. The privacy-enhanced electronic mail protocols were designed to specify security services needed to ameliorate these dangers, and to provide a mechanism for implementing those services. It will soon become a common (and widely used) protocol.

A Comment on Open Systems

by Geoff Collyer

<geoff@world.std.com>

[Editor's Note: Geoff wrote this as a reply to a comment made to another publication's request for the definition of open systems.]

I'm a system programmer, so that biases my opinions. I certainly want to be able to fix broken software. I also want the option of not running the vendor's software; not all of us are buying Application Delivery Vehicles. Changing vendors would be a good idea if there were any vendors not currently selling ADVs (I think MIPS was the last, though possibly HP still qualifies).

But my real objection to the abuse of the term "open system" has less to do with computing than with linguistic abuse. Computing is notorious for linguistic abuse and content-free buzzwords, but this case is pretty astonishing even by the lax standards of computing.

Here we have two simple English words. "System" of course became content-free years ago and now means little more than "a bag of stuff" (or as Stan Kelly-Bootle puts it in *The Devil's DP Dictionary*, "any old ratbag of incompatible components"). The word "open" has all sorts of warm fuzzy connotations of goodness (not counting Sun's innovation of Open Security, which culminated in the 386i: 'pahss-words? we don't need

no steenkin' pahss-words.'), which vendors clearly want to tap, yet it actually has denotations, notably "not closed or blocked up or sealed or locked," "not covered or concealed or restricted," and "frank, communicative" (*The Oxford Paperback Dictionary*). I am deeply and negatively impressed that Sun and SGI can call their systems "open," with straight faces, when they don't offer anything comparable to the *Digital PDP-11/70 Processor Handbook*, which described the instruction set, addressing and memory management unit, memory system, floating point processor, I/O controllers, console, kernel memory map including device addresses, and device register bit layouts in sufficient detail to implement an operating system when read in conjunction with the *Peripherals Handbook*, without "open system" hype, and all in only 276 pages.

But perhaps I am being naive and old-fashioned and should look instead at the other meanings of "open": "admitting visitors or customers," "with wide spaces between solid parts," and "not yet settled or decided" (*The Oxford Paperback Dictionary* again).

A Perspective on pathalias

by Rich Salz

<rsalz@uunet.uu.net>

[Editor's Note: A companion piece to the tenth anniversary USENET map included in this issue.]

Steve Bellovin wrote the first version of *pathalias* while working on his doctorate at the University of North Carolina at Chapel Hill. He posted it to Usenet in around 1982, and included patches to delivermail, the sendmail predecessor for BSD systems. Peter Honeyman picked it up and installed it on *allegro*, a Bell Labs machine at Murray Hill, NJ. At that time, *allegro* was the center of the UUCP mail world (a role since assumed by sites named *ihnp4*, *seismo*, and now *uunet*), and at about the same time Usenet was starting to explode all over the place. Before he left for Princeton, Peter had moved from tweaking the options to doing serious redesign of most of the algorithms. Peter is still maintaining the program, and it has followed him to the University of Michigan.

Pathalias is a very impressive piece of software engineering, and involves lots of interesting problems in memory usage, graphing, and hashing. If you are at all interested in these areas, you should check out the paper "*Pathalias, or the Care and Feeding of Relative Addresses*" that was presented at the Atlanta USENIX Conference in 1986.

Pathalias reads the connectivity lines from the map data and creates routes that follow the least cost. A connectivity line looks like this:

```
from_site to_site1(cost1), to_site2(cost2), ...
```

The costs are symbolic names chosen from the following table:

LOCAL	25	Local-area network connection
DEDICATED	95	High speed dedicated link
DIRECT	200	Toll-free call
DEMAND	300	Long-distance call
HOURLY	500	Hourly poll
EVENING	1800	Time restricted call
DAILY	5000	Daily poll
POLLED	5000	Daily poll (usually indicates from other site)
WEEKLY	30000	Irregular poll
DEAD	"infinite"	A "last chance" link

DEDICATED is usually used for network connections such as the Internet. Costs can also be omitted, as in

```
from_site to_site
```

and a default cost of somewhat better than DAILY is used (4000).

The values for the names are historical and come from the experiences of the early map users. After the names were determined, they juggled the numbers until they got maps that "looked right." - The numbers represent a measure of the bandwidth (modem and network speed), the monetary cost of the call, and the frequency of the connection. It turns out that the transmission speed - time spent on the phone - is not the major concern. The call setup time (getting a dialer, having the other machine answer, etc.) and the time between connections is really the major factor.

In addition to these constants, it is possible to do some math on the costs to tweak some of the values. For example, if one site calls another every three hours, that would be listed as HOURLY * 3. If a connection is particularly good or bad, the values HIGH (-5) and LOW (5) can be mixed in. For example if we call every hour but have lots of trouble getting through, then we might list the cost as HOURLY + LOW * 2 -- it's an hourly poll of real low quality. (If it turned out we only got through every other time, then we'd write HOURLY * 2.)

It's generally a mistake to mix the constants in any other way. If one site calls another whenever there is work, and also polls it in the evening, then the cost should be marked as DIRECT. A common mistake would be to add the two numbers, DIRECT+EVENING, which marks the connection as being worse than just calling every night!

A link can appear more than once. This is useful for maintaining private host data. For example, a site might have some expensive long-distance links that are listed in the maps, but it doesn't want to encourage their use. So in their posted map entry they might list the link as EVENING. In a private file that is also fed to *pathalias*, however, they could grade the connection as HOURLY * 3, for example. When *pathalias* sees a duplicate link, it uses the lowest of the two numbers. It is generally not a good idea to do this sort

of hiding too much, if only because some sites will actively try to re-route mail based on the published map data.

Because of the way UUCP works, if one site calls another the link is symmetric – each site will try to forward anything queued up on their side before hanging up the phone. Pathalias does not make this assumption, however. As a result, whenever the program sees a link from site1 to site2, it automatically adds a high-cost link going the other way:

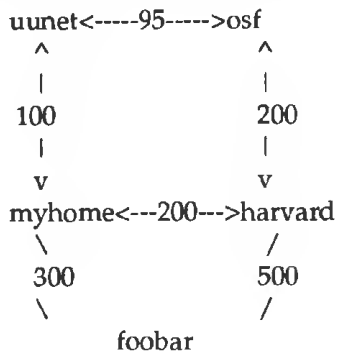
```
site2                site1(DEAD)
```

When it reads site2's map data, the cost will be set to its real value. (The map data generally is symmetric, so that links declared in one site usually have the same connectivity declared in the other site's data.)

With this in mind, we can construct a small map fragment of our own:

```
uunet  osf(DEDICATED), myhome(DEDI
        CATED+LOW)
myhome uunet(DEDICATED+LOW), harvard(
        DIRECT), foobar(DEMAND)
osf     uunet(DEDICATED), harvard(DI
        RECT)
harvard foobar(HOURLY)
```

This corresponds to the following map:



(For convenience, we've shown all links as having the same cost in each direction.) This will generate the following routes from OSF:

```
osf          %s
uunet        uunet!%s
myhome       uunet!myhome!%s
foobar       uunet!myhome!foobar!%s
harvard      harvard!%s
(The %s is replaced with the user's name.)
```

Sometimes sites do not want to act as mail relays. This is generally considered impolite. The network works as a series of exchanging favors – you carry my mail, and I'll carry yours. There are

times, however, when it makes sense to do this. For example, since ORA pays connection charges for their link to UUNET, they might not want to pass through any mail for people outside of ORA.

ORA could list their UUNET link as very expensive – EVENING or POLLED, for example. This isn't a good idea because it could result in mail taking a long, circuitous route to reach there, when in fact uunet!myhome is the shortest path.

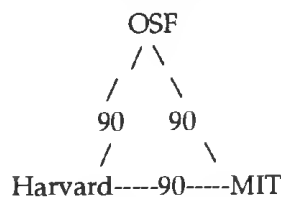
The solution is to declare a link as *terminal*. This attaches the regular cost to the link for direct connection, but assigns a very high penalty to any further link. To declare a link terminal, put the name in angle brackets. For example, if we change the uunet line in our map data from above to read like this:

```
uunet          osf(DEDICATED),
<myhome>(DEDICATED+LOW)
```

then pathalias will generate these routes:

```
osf          %s
uunet        uunet!%s
myhome       uunet!myhome!%s
foobar       harvard!foobar!%s
harvard      harvard!%s
```

Sometimes a cluster of sites are all directly interconnected. For example, MIT, Harvard, and OSF are all on a NearNet, an NFSNet regional network. The sites are all on the NearNet backbone, which uses 10 megabit/second microwave ethernet, so the cost could be listed as DEDICATED+HIGH (a high-quality network link). Pictorially, this would be:



In the map data this would show up as:

```
osf          harvard(DEDICATED+HIGH), mit(
        DEDICATED+HIGH)
mit          harvard(DEDICATED+HIGH),
osf(DEDICATED+HIGH)
harvard     osf(DEDICATED+HIGH), mit(DEDI-
        CATED+HIGH)
```

As a shorthand, we can declare a network. This is a short way of assigning single cost and full connectivity to a group of hosts. The syntax is:

```
networkname = { list of hosts } ( cost )
```


for example:

```
nearnet-backbone = { osf, harvard, mit }(DEDICATED+HIGH)
```

A common use of network declarations is for a local LAN, where all machines within a department are closely coupled.

In many instances only one machine in the group has outside UUCP links. In this case, naming the other machines in the group doesn't really serve much purpose. It would be better to do one of three things. First, arrange for the gateway to know about all the users on all the machines, and set up the mail system there so that all mail coming from any machine in the group looks like it came from the gateway. Then the other hosts don't have to be listed.

A second alternative is to declare the network as DEAD. This is usually done on the pathalias command line by using the '-d' flag. This marks all the member-to-member links as DEAD, so that the only way into a network is through a gateway. The key to this is to leave the gateway host out of the network. For example, if we want all NearNet traffic to pass through OSF, we could do the following:

```
nearnet = { harvard, mit, think, ...}(DEDICATED + HIGH)
dead { nearnet }
nearnet                osf(LOCAL)
```

The "dead { something }" syntax has the same meaning as the "-d" flag on the command line. Its advantage here is that it appears in the map data, and doesn't require all pathalias users to specify a special flag.

The best thing to do, however, is to set up a domain for the group and set up the gateway machine to be the gateway into the domain. As we [saw/shall see] there are technical requirements for a domain. Pathalias has its own definition of what a domain is, however, and it is important not to confuse them. In particular, don't use a domain address in your pathalias database unless it really is a valid registered domain.

A domain in pathalias input is a host or network whose name begins with a period, such as .COM, or .MIT.EDU. When pathalias sees such a name it automatically declares the entry as DEAD, so that it is impossible to reach a domain unless there is a gateway. For example:

```
osf                .osf.org(4000)
```

```
uunet                .uu.net
```

These two lines declare that the UUCP sites osf and uunet are gateways into the OSF.ORG and UU.NET domains, respectively. The first line shows the default cost just for explanation purposes, and to emphasize that these are standard connectivity lines, even though they look kind of different. In the real maps, however, the cost is left out and the second format is used.

The second thing that is special about a domain is that its name is appended to the end of the connecting host. Finally, when used in a network connection the domain name IS used. That is, it is appended to all the members of the network. (In a regular network declaration, the network name is just a placeholder, mostly used to find the gateway host.)

We can also have sub-domains, which are domains that are "connected" only to another domain. Putting this all together, we can get the following:

```
myhome                .com
.org                  = { .osf, .mcnc }
.osf                  earth
```

The first line indicates that myhome is connected to the .com domain. The second line sets up .org as a network, with the domains .osf and .mcnc; because the name of the network is a domain, it is used as part of the name of all the hosts in the network, so the names really become .osf.org and .mcnc.org. The third line says that .osf has a link to the machine earth. Since domain names get appended, rather than joined with !-links, we would get the following path output if we ran this map data with the local host set to myhome:

```
myhome                %s
.com                  %s
earth                  earth.osf.org!%s
```

Summary

The pathalias program is a mature mechanism for routing UUCP mail through the complex set of interconnects that comprises most of the non-Internet links of the informal 'USENET'. Its source code (on ftp.uu.net in /networking/mail/uucp/pathalias) provides interesting reading for its optimizations of some of the computationally complex parts of solving a general optimization problem.

Distributed Multi-Media: The Bandwidth Bandit

by George V. Neville-Neil
Digital Equipment Corporation
Western Software Lab, Palo Alto, CA
<gnn@wsl.pa.dec.com>

Introduction

The emergence of distributed multi-media systems, which deal with streams of audio and video data, presents several problems for applications developers and network managers. These systems must transfer large amounts of information and impose real time requirements for presentation of this data to users. For this article, I profiled TCP/IP throughput on the local network where I work for several working days to see how the available bandwidth changed over time.

Bandwidth Needs

Multimedia applications use audio and video data streams. These data streams are continuous and their presentation is time constrained. Because of this, they can put severe demands on the network. The bandwidth needed by these applications depends on the rate at which data needs to be moved and the size of each frame of data.

There are several data sizes for audio data; they are classified by their sampling frequency. Voice quality audio is the low end, made up of one byte samples, sampled at 8KHz. This translates into an 8,000 byte per second data stream. Voice quality audio is the kind of sound that you expect from a good telephone connection. At the high end of audio sampling is DAT (Digital Audio Tape) quality sound which is two channels of sixteen bit samples taken 48,000 times per second. This translates into a data stream of 192KB per second. In between these two extremes are mono and stereo quality streams that are sampled at eleven and twenty-two kilohertz. Compact Disc quality sound is two channels of sixteen bit samples, sampled at 44.1KHz, which produces a data stream of 176.4KB per second. ADPCM, which does compression, can significantly reduce the amount of data that needs to be transmitted for many applications. It can deliver certain audio streams sampled at 16KHz using only 8KB per second of bandwidth.

Video streams are harder to characterize because they often use compression algorithms. One 320 by 240 pixel frame of video data with eight bits of color information per pixel takes up about 77KB

in raw form while a full workstation screen, 1280 by 1024 eight-bit pixels is 1.31MB in raw form. NTSC, the American television standard, presents thirty frames per second which equates to a data stream of about 2.3 megabytes per second. PAL and Secam, formats commonly used in Europe, present data at twenty-five frames per second producing a 1.9 megabyte per second data stream. There are several compression techniques used to squeeze this information down into a more usable form. Algorithms that do running compression do not generally differentiate among frames, while some algorithms are applied on a frame by frame basis.

Compressing each frame using JPEG can, on average, produce a frame that is only six to eight kilobytes. At thirty frames per second this will produce 240K of data per second. The amount of JPEG compression depends on the amount of contrast variance in a frame and can produce frames that are as large as 32K. A common way of further reducing the data transmitted is to transmit fewer frames. Transmitting fifteen frames per second results in an average data rate of 120K per second. This is a form of temporal compression. Of course, the images reconstituted from compressed data have slight, usually undetectable, changes. This kind of compression is not suitable for storage of, say, payroll data.

The p.64 standard (CCITT H.261) will compress video and audio data down to a guaranteed data rate, which is expressed as p times sixty-four kilobits per second. Values of p are allowed to go from one to twenty-three. Commonly used values are one, two, and six, which correspond to one or two ISDN B channels, or a fractional T1 line. A larger value for p improves the picture and sound quality but also increases the amount of bandwidth needed.

MPEG, like p.64, is a running compression algorithm that integrates audio and video data into one stream. For a 320 by 240 pixel image at 30 frames per second, it will guarantee a data rate of 1.2Mbits per second, a little more than one tenth the theoretical bandwidth of an ethernet.

Both MPEG and p.64 use several methods to compress data using DCT transforms plus entropy coding, chroma subsampling, and inter-frame coding with motion vectors. While these methods give excellent results in compression, they also lose the frame information. The data can not be

understood as a collection of frames, but only as a stream of bytes to be uncompressed.

Applications

This article discusses three common types of distributed multi-media applications. The first application is a telephone where all of the data goes through the net. This requires two voice quality data channels, one originating at each caller. The available throughput must then be 16KB per second. Conference calling complicates this unless the network supports sending a single packet to multiple hosts (multicast). If multicast is not available then $Q(n(n-1)/2)$ data channels are needed, where n is the number of persons involved in the call, and Q is the number of data channels for full duplex communication. This means that a four person teleconference will need $2(4(4-1)/2)$, or 12 channels. If each channel is voice quality audio the entire conference will require 96KB per second bandwidth. Multicast simplifies conference calls since only one data stream is needed for each participant. Bandwidth needs can be reduced if all of the participants communicate with a central server, thereby creating a star. In this case you would need n channels, so that a four person teleconference would require only eight data streams with a total bandwidth requirement of 64KB per second.

A video lecture, where one person broadcasts image and voice to a group of others, requires at least one video and one audio channel. If this application uses voice quality audio and JPEG compression while sending fifteen video frames per second, it will produce 128KB of data per second. If multicast is not available it will require P times that data rate where P is the number of people viewing the lecture.

A final application to consider is that of a two person teleconference. This application will require two video channels, one for each direction, and two audio channels. If the video is again compressed with JPEG and transmitted at fifteen frames per second, and the audio channels are both voice quality the total data rate will be 256KB per second. If multicast is not available the number of channels required for video teleconferencing will be the same as in the telephone application except that Q will now be four, two voice and two video channels per caller.

Available Bandwidth

I measured the TCP bandwidth between my machine and three machines elsewhere within my group to get a feel for the bandwidth that was available on a typical local area network. Two of

the target machines belong to people with whom I work frequently – for whom video-conferencing would be useful. The third machine was directly adjacent on the ethernet to my machine, giving me the best possible network performance. I also measured my machine's internal bandwidth, by using localhost as a target host, to determine the maximum possible bandwidth. All of the machines were running Ultrix 4.2a and were being used for mail, news, and software development. The measurements were taken every three hours starting at 00:30 every morning from Monday, October 5th through Thursday, October 8th, 1992.

The average throughput was 597KB per second within my machine, 455KB/sec for the machine directly adjacent to mine, and 420KB/sec for the machines on my colleagues' desks. For the applications in the previous section this translates into twenty-five voice calls, three people watching a lecture, or one video teleconference.

Conclusions

For these reasons, resource allocation policies must take the human being into account. Even small glitches in an audio stream are noticeable and annoying to a person, whereas dropping the frame rate of the video is less intrusive. In all applications involving video and audio streams the audio must take priority.

Clearly, using TCP/IP over an ethernet is not the optimal transport for multi-media applications. A better transport would involve faster networks and protocols that are tailored to the special needs of these high-bandwidth, low-latency applications. Distributed multi-media must be a good neighbor on the network if it is to coexist with NFS, Telnet, and FTP traffic. Using intelligent resource allocation policies will keep applications from over-extending the network. This combination approach will make it possible to deliver distributed multi-media systems on current hardware platforms and have them safely co-exist with already installed network applications.

Acknowledgments

I would like to thank Dave Arnold, Marty Picco, Burkhard Neidecker-Lutz, and Rich Hyde for allowing me to tap them for information on video data compression rates and current standards. I would also like to thank Dave Arnold and Steve Schneider for allowing me to run these benchmarks against their machines. Finally, special thanks go to Richard Golding for ruthlessly editing this article into a clear and concise whole.

Modula-3

by Paul Vixie
<vixie@pa.dec.com>

Have you seen Modula-3?

Consider:

```
MODULE Main;                (* hello1.m3 *)

IMPORT Stdio, Wr;

BEGIN
    Wr.PutText(Stdio.stdout,
               "Hello, world.\n");
END Main.
```

The above program is written in Modula-3, one of the latest languages to descend from Algol. As you can see, the source looks a lot like Modula-2 or Pascal (or, if you live on the bleeding edge, you might agree that it looks a lot like Oberon).

As is true of the "Hello, World" program when written in other languages, our example does not begin to explicate Modula-3's interesting features; however, as with C, if you started by looking at a program which used all or even some of the interesting features, you would probably goggle and turn the page.

Modula-3 has explicit boundaries between modules. Types, constants, variables, and procedures declared in a module are only accessible to other modules if the declaring module explicitly "exports" them. (Contrast this to C, where all top-level names in a source file are globally visible at link time unless they are declared "static.") A module which depends on entities from some other module must explicitly "import" the symbols it needs.

Consider:

```
MODULE Main; (* hello2.m3 *)

FROM Wr IMPORT PutText;
FROM Stdio IMPORT stdout;

BEGIN
    PutText(stdout, "Hello, world.\n");
END Main.
```

In the first example, we imported the symbol tables from the Wr and Stdio modules, and then used their PutText and stdout symbols by qualifying those symbols with the names of the modules that exported them. In this second example, we imported only the PutText and stdout sym-

bols -- now we can refer to them without module qualifiers. (Note that the current recommended practice among experienced Modula-3 programmers is always to import symbol tables and use qualified references; you will rarely, if ever, see local symbol aliases used in published Modula-3 source.)

Modula-3 is a strongly typed language whose loopholes are difficult enough and inconvenient enough to use that one tends to localize them to one or only a few "modules" (which, for now, you can think of as "source files"). This leads toward that ideal state of affairs where most of the ugly and nonportable code in a software system is located in one place; modules which are not declared to the compiler as "able to contain ugly, nonportable code" are prohibiting from containing such code. Examples of such code include most kinds of type casts, pointer arithmetic which could lead to alignment or byte-order gaffes, and use of dynamic memory in ways that could lead to memory leaks. No one has yet invented a compiler which rejects code that is *portably* ugly.

Modula-3 has "objects," which for the OO-impaired are basically "records" (think "structs") which you can "subclass" (make another object just like this one but which adds some fields at the end or overrides the default values of existing fields) and which can contain "methods" (fields which are procedure, or function pointers, calls to which are automagically rewritten by the compiler to pass as the first parameter the object variable whose method is being referenced).

Modula-3 has "garbage collection" which means that you almost never "free" dynamic memory when you are done with it; it is reclaimed by the runtime system at some time after the last reference to it disappears. In fact the DISPOSE procedure needed to deallocate dynamic memory is only available in modules which are declared to the compiler as UNSAFE (meaning that they are allowed to contain ugly, nonportable code).

Modula-3 includes a defined "threads" interface, which means that coprocessing programs can look like what they do instead of being twisted around a very busy select(2) or poll(2) system call. The freely-available uniprocessor implementation of Modula-3's threads package uses select(2), setjmp(2) and longjmp(2), but programmers don't have to think in terms of these details and the resulting improvement in readability – and writability – of coprocessing programs is quite dramatic. And of course, if you use threads in your code and someday move it to a multiprocessor with thread support, you'll get multipro-

cessing and concurrency for free.

In support of threads, Modula-3 has mutex semaphores and a built-in LOCK statement to lock and unlock these mutexes around critical sections of code, as well as a general interface that lets threads share access to "condition variables," which BSD kernel hackers will recognize as "like sleep() and wakeup(), only how do they do that in user mode?"

For example, Figure 1 shows a short cycle burner that will prove that the thread scheduler is indeed preemptive:

```
IMPORT Stdio, Wr, Fmt, Thread, Text, Time;

TYPE
  LockedIndex = MUTEX OBJECT
    index := 0;
  END;

VAR
  Inner := NEW(LockedIndex);
  Outer := 0;

PROCEDURE InnerF(self:
  Thread.Closure): REFANY RAISES {} =
BEGIN
  LOOP
    LOCK Inner DO
      INC(Inner.index);
    END;
  END;
END InnerF;

BEGIN
  EVAL Thread.Fork(
    NEW(Thread.Closure,
      apply := InnerF));
  LOOP
    Time.LongPause(1);
    LOCK Inner DO
      Wr.PutText(Stdio.stdout,
        "Inner=" &
        Fmt.Int(Inner.index) & "; " &
        "Outer=" & Fmt.Int(Outer)
          & "\n");
      Wr.Flush(Stdio.stdout);
      INC(Outer);
      Inner.index := 0;
    END;
  END;
END Main.
```

Figure 1: MODULE Main

This program forks a thread which increments, with locking, a global index variable. We made this variable an "object" which subclasses the "MUTEX" object type, since this is the usual style for object type with a single mutex to lock all of its resources. We could as easily have made this a

"record" type with an explicit mutex field; for that matter we could have made the index and the mutex separate global variables with no "record" or "object" type to aggregate them. Anyway, the main thread forks a thread that executes the ApplyF function, which loops forever, incrementing the global index variable which is protected by a mutex. The main thread then loops, waiting one second and then prints and clears the index that the other thread is furiously incrementing. On the author's workstation, this program prints:

```
% ./a.out
Inner=144204; Outer=0   Inner=126392;
Outer=1                 Inner=114215;
Outer=2                 Inner=125996;
Outer=3
^C
```

Exceptions

Last in our brief survey of Modula-3's features, we note "exceptions." An experienced C programmer knows that there are two kinds of code: the kind in which you check the return codes of all system and library calls, and the kind that people actually write.

In the first kind (which has been characterized as "you're crossing the street to buy an ice cream cone, so after every step you stop and check yourself all over to see if you have just been hit by a crashlanding 747 jetliner"), you discover that return codes breed more return codes, since if your function discovers an unusual error it would like to propagate this unusualness to its caller, which must do the same, all the way up the call stack until some caller "handles" it (or more often in the code we've seen, casts it to "void"). Modula-3 has, as part of every procedure declaration, a list of "conditions" which that procedure is capable of "raising." Code which calls that procedure has the option of wrapping the call in a TRY...EXCEPT...END statement if it wants to have a chance to "handle" certain exceptions; in the absence of any caller who cares, the program exits with an error. This leaves the return value available for a full range of useful things, none of which are reserved as magic cookies. It also avoids most occasions where an error encoding must be mapped from "-1 was an error" to "but NULL is my return error code."

A Larger Example

So, why would you care about Modula-3, given that the world seems to be switching to C++ and Objective-C or pouring megaspecmarks into Common Lisp? Simply put, "it ain't over 'til the fat lady sings." C++ is an endlessly flexible lan-

guage, much worse than C; measured by the ability to write code which other programmers cannot understand or to write code which even the author cannot understand or to write code which does not do what it looks like it should be doing, C++ is the first language to eclipse C. (Sorry, Larry, C++ beat Perl by a couple of years.) One could (and many have) argued that C++'s design not only permits bad code, it encourages it. Common Lisp, on the other hand, is a beautiful, elegant language which will someday join ADA in the museum of beached whales.

Of the languages whose definitions are in the public domain and for which there are freely-available portable implementations available for most of the popular POSIX-like platforms, Modula-3 is the one in which it is hardest to write code which does not do what it looks like it should be doing, or which even the author cannot understand, or which other programmers cannot understand. You may not see this from the examples shown so far, but consider the program in Figure 2, which sends an IP/UDP datagram (think "packet") requesting the network time from all `inetd(8)`'s on the local subnet, collects the replies, and prints the average or "network" time.

```

IMPORT Datagram, Netdb, Net, MyTime;
(* local hacks *)
IMPORT Stdio, Wr, Fmt, Thread, Time,
Text; (* standard *)
IMPORT Word, Ctypes; (* nonportable *)

TYPE
  T = Ctypes.long; (* 64-bit machines
will in be trouble here *)

VAR
  Done:=FALSE;
  Rcvd:=0;
  Waits:=0;
  NetTime: T;

CONST
  TimeDifferential = -2085978496;
(* inetd(8)'s offset *)

PROCEDURE ProtocolF(self:
  Thread.Closure):
  REFANY RAISES {} =

VAR
  port := Datagram.NewClient(NIL,
  mayBroadcast := TRUE);
  server := Netdb.NewRemotePort(
  "255.255.255.255", "time", "udp");
  timeval := NEW(REF T);
BEGIN
  port.send(NIL, 0, server);

```

```

(* 0-length datagram is a "request" *)

LOOP
  EVAL port.recv(timeval,
  BYTESIZE(timeval^), server);
  timeval^ := Word.Minus(
  Net.ntohl(timeval^),
  TimeDifferential)
  IF Rcvd = 0 THEN
  NetTime := timeval^;
  ELSE
  NetTime := Word.Divide(
  Word.Plus(NetTime,
  timeval^), 2);
  END;
  INC(Rcvd);
  Done := FALSE;
END;
END ProtocolF;

BEGIN
  EVAL Thread.Fork(NEW(Thread.Closure,
  apply := ProtocolF));
  REPEAT
  Done := TRUE;
  Time.LongPause(1);
  INC(Waits);
  UNTIL Done;
  IF Rcvd = 0 THEN
  Wr.PutText(Stdio.stdout,
  "No Replies.\n");
  ELSE
  NetTime := Word.Minus(NetTime,
  Waits);
  Wr.PutText(Stdio.stdout,
  "Network time: " &
  MyTime.TimeToText(NetTime) &
  " (" & Fmt.Int(Rcvd) & "
replies)\n");
  END;
END Main.

```

Figure 2: UNSAFE MODULE Main

One important note about this code: the Datagram module is a quick and dirty hack that we cobbled together to test some assumptions about IP/UDP performance; the actual IP/UDP interface supported in Modula-3 is likely to be quite different. Likewise the Netdb, Net, and MyTime modules are all local hacks that you don't have and wouldn't want anyway. As is true of any language which is less than a decade old, Modula-3's standard libraries are still evolving.

The program makes slightly contrived use of the Thread interface; the goal is to keep collecting responses until none appear for one second. A C programmer would use `alarm(2)` or `select(2)` with a timeout. This program starts a thread which blocks in `port.recv()` (which, given the presence of the Thread interface, was designed without any

explicit timeouts of its own); whenever a datagram comes in, this thread receives it and computes it into the running average. The main thread loops, waiting one second and then exiting the loop only when no datagrams have been received by the other thread during the last second. The code is sloppy in that it should protect its thread-shared variables with mutexes, but as a demonstration it is already as complicated as would be useful.

The program is also of the “ugly and nonportable” variety; a more robust implementation would hide all of the details of the Word (“unsigned”) arithmetic in other modules so that this module could do its job in as straightforward a manner as possible. We chose this example because it shows Modula-3 code trying to deal with the UNIX system call interface. This, in other words, is as ugly as system-dependent Modula-3 source ever has to get. You might wonder why we NEW() the timeval variable and dereference it everywhere rather than creating a normal variable and passing it by ADR() in the one place we actually need its address. This has to do with the declaration of the Datagram object’s recv method, which due to dampness behind its authors’ ears, was rather more selective than it could have been.

To get an idea of the real possibilities given threads and garbage collection, consider an IP/DNS name server which needs to process concurrently both multiple incoming and outgoing “zone transfers” over IP/TCP, all the while receiving, forwarding, and generating DNS requests and replies over IP/UDP. The popular BIND name server forks a subprocess for each zone transfer – a major performance penalty if you don’t have a copy-on-write fork() and your nameserver core image is tens of megabytes in size. BIND also has a very busy select(2) at its core, along with a memory management scheme that can make grown programmers want to quit their jobs and go drive tow-trucks. Given garbage collection and threads, the hardest parts of this sort of program just obviously slide into place with almost zero insertion force. Any C programmer who likes to use the CPP to layer garbage collection on top of malloc(3) and threads on top of select(2) will probably not enjoy Modula-3 very much since all that stuff is done for you and your application’s code is mostly goal- rather than mechanism-oriented.

Comparisons

The features highlighted by the last example are: (1) variables can be given types, or initial values, or both, and if both are specified then the initial

value must be of the given type; (2) this is also true of formal procedure parameters; (3) actual procedure parameters may be given positionally or by name; (4) aggregates may be returned by functions, or assigned to local procedure (“auto”) variables; (5) if you want to call a typed procedure (“function”) and throw away the result, you have to explicitly EVAL it; (6) EVAL in Modula-3 does not do anything like what it does in Lisp or Perl; (7) expressions of type TEXT, which includes “quoted strings” and results of functions of type TEXT, can be catenated inline with the “&” operator; (8) the ever-present newline (“\n”) works as you’d expect; (9) most statements are innately compound, which means that IF and WHILE need an END, but BEGIN is meaningless for them; (10) dynamic variables created with NEW can be forgotten about, with no explicit deallocation; (11) expressions can be contained in parenthesis but need not be, and the expression used for IF and WHILE and REPEAT can be parenthesisless.

Features which are not highlighted in this example but which are interesting: (1) (*comments (*can be*) nested*); (2) NEW can fill in or override default values of fields in record (“struct”) or object (“*struct with magic*”) types; (3) record and object fields can *have* default values, which, as with variables and formal procedure parameters, cause the type to be imputed if no type is specified.

Differences from C which you will probably find bizarre or irritating: (1) NEW takes as its argument a pointer type rather than the size or type of the thing being allocated; (2) there are no (pre-, post)-(inc,dec)rement operators, so you have to use INC(x) for x++ and DEC(x) for x--, and neither INC() or DEC() returns the new value; (3) arithmetic on unsigned integers is painful and awkward; (4) compilation time and object size are both very large compared to PCC or GCC if you use the current version of the freely-available DEC SRC compiler (*which is the only one in existence at this time*); (5) case is significant even for built-ins, so you *must* type a fair amount of your program text in UPPER CASE; (6) printf() is not impossible but not straightforward, either.

Differences from C++ which will either make you tense or relieved: (1) there are no enforced constructors or destructors, and though there is a convention for constructors there is none for destructors; (2) multiple inheritance, long considered either a botch or a blight (depending on who you ask), isn’t here at all.

The advantages (or disadvantages, depending on

who you ask) of strong type checking have already been argued elsewhere. To the oft-quoted "strong type-checking is for weak minds" argument, we counter that "software systems are getting larger and more complex; programmers' minds are not."

History

Modula-3 was designed in the late 1980's by researchers at Digital's Systems Research Center ("DEC SRC") in Palo Alto, California, and at the Olivetti Research Center ("ORC") in Menlo Park, California. It descends most recently from Modula-2+, which came from Cedar and Mesa and Pascal; Modula-3 was lightly cross-pollinated with Oberon, as Niklaus Wirth (creator of Pascal, Modula-2, Oberon, and more recently Oberon-2) was on sabbatical at DEC SRC during part of the time that Modula-3 was being conceived. Legend has it that Wirth's main contribution to Modula-3 was to encourage its designers to leave things out; this "smallness" is apparent in that Modula-3 is smaller by far than Modula-2+, though it is still larger than Oberon or Oberon-2.

A portable implementation of Modula-3 was written at DEC SRC and has been made more-or-less freely available by Digital Equipment Corporation. This compiler generates C as its intermediate language, which accounts not only for its portability but also its moderate speed and largish object code size; on the bright side it is free, and runs on most of the common POSIX-like platforms including Ultrix (MIPS and VAX), SunOS (SPARC and 68000), RS6000, and HP-UX (PA-RISC and 68000). More ports are under way, as is development and standardization of the runtime library. There are restrictions on the use of this compiler for commercial products, but you should get those details by reading the release notes that come with the DEC SRC compiler.

Future

To many programmers, C and C++ seem like foregone conclusions. Modula-3 is the only serious challenger to C++ as the next massively popular system and application programming language. This author believes that there is a good chance that there will be a market for programmers and CASE tools in Modula-3 in the next year or two, since it is a practical yet elegant software design and implementation tool which encourages clean, bug-free code and is a true pleasure to work in no matter how large the project might be (or become).

If you enjoyed Modula-2 or Pascal but found them incomplete and limiting, it's a safe bet that you will find whatever you were missing...in Modula-3.

If C is driving you nuts but you get cold sweats whenever you think about C++, you may find a way out of your dilemma...in Modula-3.

Resources

"System Programming with Modula-3", Greg E. Nelson et al, Prentice Hall

"Modula-3", Samuel P. Harbison, Prentice Hall

<comp.lang.modula3> (usenet newsgroup)

m3@src.dec.com or decwrl:m3 (mailing list, gatewayed to comp.lang.modula3)

gatekeeper.dec.com:~ftp/DEC/Modula-3/* (a freely available, portable Modula-3 compiler for several UNIX variants including Ultrix, SunOS, HP-UX.)

An Update of UNIX-Related Standards Activities

by Stephen Walli

Report Editor

<stephe@usenix.org>

USENIX Standards Watchdog Committee

Report on The IEEE Standards Board

Mary Lynne Nielsen <m.nielsen@ieee.org> reports on the June 1992 meeting:

The meeting was the occasion for the approval of two more POSIX standards and further activity concerning IT standards in general.

NesCom and RevCom Actions

Two TCOS standards were before the IEEE Standards Board Review Committee (RevCom) for approval as IEEE standards at this meeting – POSIX.5, the POSIX Ada Language Binding to IEEE Std 1003.1-1990 (ISO/IEC 9945-1: 1990), and POSIX.9, the POSIX FORTRAN 77 Language Binding to IEEE Std 1003.1-1990. Both were approved straightforwardly, which is a credit to their chairs for completing the difficult work involved in coordination.

One lesson to be learned from their experiences – RevCom requires that the names of negative balloters be attached to each negative ballot when these objections are submitted with the RevCom package. It was a bit of a scramble for the chairs to come up with the appropriate documentation. Chairs should ensure that their records clearly reflect committee actions.

The IEEE Standards Board New Standards Committee (NesCom) also had a revised Project Authorization Request (PAR) for POSIX.5 on its agenda. (Seems they had never revised its original PAR, which said it was doing an Ada binding for all of POSIX!! Don't want to imagine the size of that document!) This PAR had been lost in the shuffle for awhile, but NesCom agreed to consider it at the same time as RevCom, in an exception to their rules. It was approved straightforwardly.

The unapproved PAR for POSIX.19 (the Fortran 90 binding to POSIX.1) remained unapproved, as the working group did not explain its relationship to the X3 Fortran committee in a satisfactory manner to NesCom. This will appear on the NesCom agenda again in September.

Congratulations all around to those folks involved in POSIX.5 and POSIX.9. Developing a consensus standard is a long and painstaking process, and everyone deserves a great deal of credit for finally getting there!

The most wide-ranging actions that affect TCOS, however, occurred in groups other NesCom and RevCom.

IT Funding

The Standards Board had created an ad-hoc committee in March to look at the issue of funding of Information Technology (IT) activities. The American National Standards Institute (ANSI) had made a proposal that the cost of involvement in international standards development in the IT area be covered by the individuals involved in those activities. This would mean that anyone involved in these standards would be charged a fee to cover the administrative costs that ANSI incurs as the secretariat to JTC1.

As the IEEE is a major developer of standards in this arena, the subject concerned the Board greatly, and in March an ad-hoc committee was appointed to review the issue. At the June meeting, the committee reported that it recommended that interim support be given to the JTC1 secretariat contingent to the IEEE receiving a seat on the committee that oversees this involvement. It further recommended that professional opinions be obtained as to the legal, financial, and tax implications of IEEE committees being assessed for the financial support of ANSI secretariats. The final report of this committee is expected in September.

One note: this subject was discussed in great detail at the TCOS Standards Executive Committee (SEC) meeting in July, and a motion was passed that recommended general support while encouraging involvement of IT standards developers in any final decision. This resolution has been forwarded to members of the Standards Board ad-hoc committee as a contribution.

Other board news involved reports on the JTC1 TAG (Technical Advisory Group, the US national member group to JTC1). The IEEE had voted "no" on the proposed merger of X3 and the JTC1 TAG, which had been proposed in several forms for the past six to nine months. The proposal for this

merger has now been dropped. Changes to the JTC1 TAG procedures were recommended from the meetings on this issue, however, and those are expected to be developed in the future.

The JTC1 TAG also authorized three simultaneous ballots in the IEEE and in the JTC1 TAG of P1224, P1224.1, and POSIX.17. This is a ground breaking process that should result in faster advancement of these standards into the international arena.

Finally, the IEEE Standards Board Procedures Committee (ProCom) took action on the ongoing requirement for approval letters from companies to include company acknowledgments in a standard. ProCom, after approving this process last December, voted in June not to include such company acknowledgments in standards.

ProCom felt that the policy of obtaining a letter of permission from each company still allowed the possibility that the person writing the letter was not the appropriate person to authorize the acknowledgment. In addition, there was no equitable way of acknowledging everyone associated with the standard by having some companies send in letters and some not. As such, ProCom felt that it was simpler not to include company acknowledgments at all.

The only problem with this, of course, is that ProCom announced this policy and began to implement it just six months ago. Many groups have begun to do all the leg work involved in getting letters signed by the appropriate personnel in their departments, and those letters have been coming into the IEEE. As such, ProCom made a somewhat awkward policy change, which only exacerbates the perception that "they're always changing the rules."

ProCom was well aware that this perception could exist, and discussed various ways to try to record their rationale for such changes. Nevertheless, they felt the implications of this policy were too unsettling to allow it to continue for a longer period of time.

By the way, this series of Board meetings was the first held outside of Regions 1-6. Region 9 hosted this meeting in San Juan, Puerto Rico. The Board was received enthusiastically, and the week was devoted to extra sessions and inclusions of special seminars. This was a result and a reflection of the IEEE's worldwide membership and was a large success.

Report on POSIX.6: Security Extensions

Charisse Castignoli <charisse@Smallworks.com> reports on the July 13-17, 1992 meeting in Chicago, IL:

The POSIX.6 group continued to work on new project authorization requests (PARs). Two PARs have been submitted to the Project Management Committee (PMC). They are:

- A Secure General Terminal Interface (GTI)
- Identification and Authentication

Other PARs, such as a portable interchange format, have not been submitted to the PMC due to the lack of resources to work on them.

In response to requests by individuals to assess whether or not POSIX.6 could go out as a trial use standard, Mike Ressler suggested we carefully analyze this approach. We need to go back and look at what the Trial Use definition from the IEEE is, and try to determine what the right approach is for POSIX in general, NOT just POSIX.6.

Monday:

The POSIX.6 ballot resolution committee continued to slog through the comments and objections. We work individually on our laptops, and then send a merged document back to Bellcore. Mike Ressler and his horde of great editors then patch together our individual sections and email out the updated sections.

Of course, you can imagine what happens when a laptop breaks down. The person depending upon it instantly becomes an order of magnitude less productive. This week's session began with the power supply failure of one of the laptops. No problem, says customer service, we'll ship you a new power supply overnight and have you up and running in no time. We should have started taking odds on whether the power supply or the end of the meeting would arrive first!

Despite our hardware limitations, the committee still struggled on....

Tuesday:

We spoke for a long time about multi-level directories and whether or not they should be in the standard. Multilevel directories, are a technique used to solve the problem of public directories (such as */tmp* and */usr/spool*). In a trusted system with more than one sensitivity level, a process at SECRET cannot view files created by processes at TOP SECRET. To solve this problem, the idea of creating non-visible subdirectories, one for each

level, was hatched. Processes without a privilege will only see files in their subdirectory. To this process, the pathname would look like */tmp/mysecretfile*. But to a process with the multilevel privilege the pathname would look like */tmp/SECRET/mysecretfile*.

Kevin Brady pointed out that there were very few existing applications that actually needed to view the resulting true multilevel directory. Most applications just want to create a file and are unaware of whether the underlying directory is multilevel or not.

For example, in current UNIX trusted systems, vi writes file to */tmp*. vi is unaware that */tmp* is a multilevel directory, however *expreserve*, the program that reclaims vi drafts from */tmp*, is multi-level aware.

Our power supply didn't arrive today....

Wednesday:

One of the most controversial ballot resolution issues we face is that we do not have a consistent storage and allocation model for the data structures that the POSIX.6 interfaces manipulate. Some functions, such as the Mandatory Access Control (MAC) and Information Labels (IL) interfaces, lend themselves to persistent opaque data types. Others, such as the Access Control List (ACL) interfaces, require data types that are non-persistent.

It is amazing that almost two years after this issue was raised, after many hours of thought and great debates over countless beers, we reach the final hours of ballot resolution and still have not reached consensus. The resolution of the day is:

- MAC and IL are going to be persistent opaque,
- Audit, Discretionary Access Control (DAC), and PRIV are going to be non-persistent opaque

Still no power supply and it's the shipper's fault according to customer service.

Thursday:

The next controversial issue that was raised has its origins even further back in the history of POSIX.6. The discussions go all the way back to */usr/group* meetings! This is the ACL feature called the mask. The mask was introduced as a mechanism to:

- map UNIX mode bits into an ACL,

- map *chmod()* calls to manipulations of an ACL

- provide backwards compatibility with the current uses of the mode word.

In order to achieve maximum compatibility (but not 100%), the ACL algorithm became incredibly complex as ACL entries became subject to restrictions and manipulations incurred by the mask. The algorithm became esoteric, to the point where this reviewer believes that no one without a PhD in computer security will be able to understand it.

In order to simplify the algorithm, the mask has been deleted. Now, mode bits are converted to ACL entries, and *chmod()* only affects the UNIX mode bits. A POSIX configuration option allows the application to select whether or not to receive an error when *chmod()* is executed on a file that has an ACL on.

No power supply – but it will be there tomorrow for sure for sure.

Friday:

Most groups are 80-95% complete on their pass through the objections and comments. ACLs, who had a few extra to begin with, still have the furthest to go. The committee would like to go out for re-ballot or re-distribution at the end of the Utrecht meeting.

UPS finally delivers Roland's new power supply just in time to pack up his laptop and get absolutely no use out of it whatsoever.

Report on POSIX.17 – Directory Services API

Mark Hazzard <markh@rsvl.unisys.com> reports on the meeting in Chicago July 13-17, 1992:

Summary

Draft 3.0 of POSIX.17 completed the first round of IEEE balloting in May. We met primarily as a ballot resolution team in Chicago, resolving 98% of all outstanding comments and objections. Since the Chicago meeting, we have finished Draft 3.0 ballot resolution, and published and re-circulated Draft 4.0 for ballot. We plan to get the ballot results in time to resolve comments at the Utrecht meeting in October. From there we plan to submit the balloted specification to the IEEE for final approval, publication, and forwarding to ISO for fast tracking (i.e. direct ISO ballot).

Our Project Authorization Request (PAR) has been split/recast into 4 separate PARs to:

1. separate the Directory Services API work (which is almost finished) from the POSIX name space issue which hasn't received much attention, and
2. separate the actual document into a format aligned with ISO expectations.

Introduction

The POSIX.17 group has generated and is currently balloting a user to directory services API (e.g. API to an X.500 DUA - Directory User Agent). We used APIA - X/Open's XDS specification as a basis for work. XDS is included in XPG4 and has been adopted as part of both OSF's Distributed Computing Environment (DCE) and Unix International's Atlas.

XDS is an object oriented interface and requires a companion specification (XOM) for object management. XOM is a stand-alone specification with general applicability beyond the API to directory services. It will be used by IEEE P1224.1 (X.400 API) and possibly other POSIX groups, and is being standardized by POSIX/TCOS as P1224. A draft of P1224 is already in ballot.

Status

POSIX.17 was reviewed by the Project Management Committee for the Chicago meeting without problem.

Draft 3.0 of POSIX.17, which included all test methods and its Language Independent Specification (LIS), completed IEEE ballot prior to the Chicago. The group spent a majority of the meeting processing the results of that ballot. Over 200 comments/objections were processed, with all but four tentatively resolved. Actions were assigned to resolve the remaining four. Our technical editor did an incredible job in producing Draft 4.0 in time for a recirculation ballot, which closed October 5th.

POSIX.17 was one of three TCOS-SS projects recommended for fast track ballot to ISO during a special ad hoc meeting of the US TAG to JTC1. [Ed. - ISO is responsible for developing and approving of international standards. TCOS-SS (Technical Committee on Operating Systems - Standards Subcommittee) is the IEEE committee responsible for developing operating system standards, e.g. POSIX. Documents developed by the IEEE can be forwarded by an ANSI (hence U.S.) Technical Advisory Group (TAG) to the

Joint Technical Committee (JTC1) of ISO and the International Electrotechnical Committee (IEC) for consideration as ISO standards.]

In order to accommodate the ISO format, POSIX.17 needed to be split into four separate parts (documents). To that end, four PARs were submitted to the IEEE which requires a PAR for every document. This in effect revises our current work to reflect the ISO format requirements. These have been reviewed and accepted by the IEEE Review Committee (RevCom) and assigned the following project numbers under the general heading of "OSI APIs".

P1224.2 - Directory Services API - Language Independent specification

P1326.2 - Test Methods for P1224.2

P1327.2 - C Language Binding for P1224.2

P1328.2 - Test Methods for P1327.2

My understanding is that when P1224.2 and P1327.2 are approved by the IEEE, they will be proposed to ISO as Draft International Standards (DIS).

In Closing ...

The group is meeting in Utrecht in October, where we plan to process the results of our September recirculation ballot. If all goes to plan, we will submit our specification to the IEEE for acceptance as a standard before year's end. Based on this schedule, I would expect to see it published by the IEEE in the first half of 1993.

Report on POSIX Distributed Security Study Group

David Rogers reports on the July 13-17, 1992 meeting in Chicago, IL:

Background

In October 1991, as a result of the activities of the informal liaison group between the Security, System Administration, and Distributed Services working groups, a draft project authorization request PAR was circulated for discussion. This draft PAR proposed a working group to define a POSIX.0 model for security in a distributed system, and the definition of security interfaces in a distributed environment.

From the discussions on the draft PAR, a study group was proposed to investigate the subject more thoroughly and, if appropriate, produce a more clearly defined PAR.

As a consequence, a BOF was held at the January 1992 POSIX meeting and the formation of the Distributed Security Study Group under the auspices of the Distributed Services Steering Committee was approved by the POSIX Sponsor Executive Committee (SEC).

Current Status

Two full meetings of the study group have been held, with a core of about 10 people. The initial emphasis has been to define a framework or model, based on the POSIX.0 model, in which to place the required security functionality into context and to identify suitable APIs.

The first meeting entertained a set of presentations on the OSF's Distributed Computing Environment (DCE), the ECMA SESAME project, and Secureware's MAXSIX. We wanted to review input on existing or emerging practice and architectures.

Following this, an abstract approach to develop the model was tried, based upon the POSIX.0 model. One overheard comment was that "They don't even know what planet they are headed for." However, the meeting did agree to a suggestion that the ECMA Security Framework (described in ECMA TR/46) should be used as a starting point. Additionally the GSSAPI was identified as a potential candidate for a base imple-

mentation. Accordingly, liaison was initiated with the Internet Engineering Task force (IETF) on the status of the Generic Security Service API (GSSAPI) and potential need for extensions to it.

An initial draft paper mapping the ECMA Security Framework into a POSIX.0 model with POSIX.1 and POSIX.6 was produced between the April and July meetings. The ideas in this were reviewed during the July meeting, together with the overall structure and content of the proposed report to be produced by the study group. The POSIX Security Framework document is being further developed prior to the October meeting in Utrecht.

Liaison with Other Organizations

Shortly after the April meeting it was brought to the attention of the chair of the study group that X/Open were also proposing work of a similar nature and scope, including approaching the IETF regarding GSSAPI. (In fact the IETF working group chair received approaches from POSIX and X/Open within 2 hours of each other!) Several meetings have been held between members of the study group and X/Open representatives to ensure that the respective groups coordinate their activities and do not unnecessarily diverge or conflict.

Student Stipends

The Association allocates funds to award a limited number of grants to cover accommodations and registration fees to full-time students who wish to attend our conferences and workshops. Interested full-time students who wish to apply for a grant may obtain a copy of the student grant application form by looking for periodic postings on *comp.org.usenix* or contacting the Association's headquarters. Upcoming conferences and their respective grant deadlines are:

Winter Technical Conference:

January 25-29, 1993 in San Diego, California
Grant Application due: December 15, 1992

Applications Development Workshop:

March 29 - April 1, 1993 in Toronto, CANADA
Grant Application due: March 17, 1993

MACH III Symposium:

April 19-21, 1993 in Santa Fe, New Mexico
Grant Application due: April 7, 1993

The Bookworm

by Peter H. Salus
Sun User Group, Inc.
<peter@sug.org>

X.desktop

Long ago and far away, at the UKUUG Conference in London in July 1990, Clive Feather of IXI Limited demonstrated X.desktop for me. I admit to having been quite impressed. Burgard and Moore in their *X.desktop Cookbook* have now provided an excellent guide to customizing IXI's desktop manager.

My guess is that most of the writing is Burgard's (he's a journalist) and most of the technical stuff is Moore's (he's an IXI employee), but it doesn't matter. The book is relatively easy to read and the details are more than merely adequate. Scattered through the book are boxes labeled "Hot Tip." Whoever thought of these – perhaps an editor at Prentice Hall – deserves my gold star for this month.

Obfuscated C Code

You most likely don't have to be an old-timer to know about Landon Curt Noll's Obfuscated C contests, which began in 1984. Well, Don Libes has taken the results of the first eight contests and built a book around them. Most of the volume, *Obfuscated C and Other Mysteries* has appeared as columns in the *C Users Journal*, but it's nice to have them in one place. Libes' advantages are really quite straightforward: he understands C and he can write in a humorous fashion. The book starts with a number of quite elementary chapters, but it turns into an excellent teaching/learning device in chapter 3, where the four "winning programs" of 1984 are featured. Looking at code that has been destroyed in this fashion makes good sense: it tests one's ability to understand just what lies behind the veils and it enables you to comprehend the advantages of neat – perhaps even elegant – code. [Several years ago, Marc Donner gave a course at NYU on "How to Read." He would have wanted this book for a text.]

The other chapters/columns are useful, but the ones on the various Obfuscated C contests and their winners are really excellent. This book is both fun and informative.

The Internet, again

This must be the year for Internet books: Kehoe's *Zen* and Krol's *Whole Internet Catalog* were in my October/November column. Now I've got Marshall Rose's *The Internet Message*, listed as "the fourth book in MTR's networking trilogy." It is interesting reading, but if what you want is how-to stuff, one of the other two volumes is probably better.

However, at the very end of the volume, Rose reprints his paper of last May's IFIP conference (Vancouver). This summary of Rose's ideas on where OSI has been, where it now is, and where it's going should be required reading for anyone interested in such strange and bizarre things as X.500, FTAM, and CMIP. To me, Rose's criticisms are as good as those of Malamud. Between them, they give us a lot of good reasons as to why OSI's market performance is less-than-inspiring.

On the other hand, Rose's bibliography is in order-referred-to, making it about as useful as a dozen buggy whips. What ever happened to things like alphabetical order by last name of author?

And more...

Once upon a time (last year) Carl Malamud attempted to put the ITU Blue Book on line. ITU killed the project ("Bruno") in three months. To get the whole story, run out and buy Malamud's new book *Exploring the Internet*. It is subtitled "a technical travelogue," and that's what it is. Malamud seems to have circled the world thrice in under a year, visiting lots of folks, eating in a number of restaurants I wish I could get to, and recounting conversations and naming names where the bureaucracy of international telecommunications is concerned.

It is a wonderful book, but the tale of Bruno and its slaughter at the hands of ISO and ITU may be the funniest and saddest in the book.

Malamud's book is entertaining, informative, and lots of other things. I love his anecdotal style. This is not a high-tech volume on the Internet. But it is a great tour. And it will explain why the international standards bodies don't want standards to be readily available; why they genuinely fear the Internet and the wonder that John Quarterman calls "The Matrix;" and just why OSI is less than a success, despite the support of many governments and PTTs.

Under the Christmas tree

It seems to me that I should end up this year recommending a few things about which you might want to drop hints to folks who purchase gifts at this time of year.

My number one choice is Malamud's *Exploring the Internet*. Even if you don't care about standards or about OSI, the tale of the eventually confiscated sausage is worth reading.

Next, I'd like to recommend Rich Stevens' *Advanced Programming in the UNIX Environment*. I mentioned this volume several months ago, and Peter Collinson reviewed it at length in the last issue of this newsletter so there's little for me to say. But at over \$50, it's a nice gift.

My last book-of-the-year choice goes to Ed Krol's *Internet* book (O'Reilly). Both I and Billy Barron have reviewed it in these pages, so I'll leave it at that.

Finally, a non-book: I'd like to thank Rob Pike for making the Plan 9 manuals available by anonymous ftp [from research.att.com:dist/plan9man] and to Geoff Collyer for making them available through the On-Line Book Initiative [from obi.st-d.com:obi/Bell.Labs/plan9pm]. This is something that an alert publisher should put in print soon.

Have a safe and happy holiday season. See you next year!

X.desktop Cookbook (Prentice Hall, 1993; 380pp.; ISBN 0-13-978537) \$38.00

Obfuscated C and Other Mysteries; (John Wylie & Sons, 1993; 408pp.; ISBN 0-471-57805-3;) \$39.95 for a book/DOS disk set

The Internet Message, Marshall Rose (Prentice Hall, 1993; 370pp.; ISBN 0-13-092941-3)

Exploring the Internet, Carl Malamud (Prentice Hall, 1992; 379pp.; ISBN 0-13-296898-3) \$26.95

Crossing the Internet Threshold

Reviewed by Billy Barron

<billy@unt.edu>

The latest entry into the Internet book realm is *Crossing the Internet Threshold: an instructional handbook* by Roy Tennant, John Ober, and Anne G. Lipow with a foreword by Clifford Lynch. Its price is \$40 from Library Solutions Press at (510) 841-2933. The ISBN is 1-882208-01-3. I have a pre-release version of the handbook and not the final version.

The book tries to serve two audiences at once. The first is the beginning Internet user; the other is 'Internet trainers' as a training supplement. I fall into the second category. The handbook slightly leans towards librarians, but not enough that it hurts the readability for others. It is a mixture of materials from lectures, overheads, one page summaries, exercises, and checklists.

The handbook is one of the most non-threatening documents that I have seen on the Internet. The material is very concise without needless detail, but at the same time covers all the basics and is very accurate. One reason it is concise is that it describes the best documents, periodicals, and discussion groups for those wishing more than

the basics. The material is so straightforward that I saw only one spot in the book where a user could get totally lost (the LISTSERV section), though on a second reading I realized that all the necessary steps were there and correct.

The one page summaries cover all kinds of related but non-essential topics like BITNET, Gopher, and Project Gutenberg. The exercises are well thought out. The overheads and such are of use to an Internet trainer who has not already designed similar materials.

In summary, *Crossing the Internet Threshold* met the needs of its target audiences. It is not a book for the experienced Internet users who do not train and is not presented as such. My main complaint about the book is the price. The reason that has been explained to me is that this is the only title Library Solutions Press has and all of the overhead has to be covered by this book. I think it would make a useful addition to the book collection of both neophytes and Internet trainers alike.

The Internet Companion

Reviewed by Billy Barron

<billy@unt.edu>

The Internet Companion: A Beginner's Guide to Global Networking (ISBN 0-201-62224-6) by Tracy LaQuey and Jeanne C. Ryer with foreword by Senator Al Gore is a book on the Internet. Its cost is a low \$10.95, which is half the cost of any of the other Internet books currently on the market.

The title says it is a beginner's guide and *The Internet Companion* is exactly that. If you are hoping to become a power user, then buy a copy of "Zen and the Art of the Internet" or *The Whole Internet User's Guide and Catalog* instead. *The Internet Companion* is directed to the beginning novice – especially one who is either not currently connected or not sure why they should use the Internet. It even covers beginning topics such as what files and accounts are.

A good portion of the book is spent on selling the Internet. This is largely done through the use of anecdotes, ranging from the cultural "Enough of White Man's ASCII" to the political "Serious Games" to the silly "Elvis Sighted on the Internet." They provide quick glances into all aspects of Internet life and hopefully will excite some potential users.

One of the principles used in writing this book was to only show how-to examples when the example would help (and not confuse the user

due to system differences). Therefore you will find examples of Telnet and FTP commands, which are pretty universal, but not of e-mail and USENET because there are dozens of very different mail and news packages. Since I have complained about the dependence on a single mail and news package in other Internet books, I found this a refreshing change.

A chapter called "Getting Connected" is directed at the user who is need of an individual or small business Internet connection. It explains all the major issues involved with getting connected and then goes on to list the network providers that offer this type of service. After this chapter, the book closes with a very complete bibliography.

In conclusion, if you are already a competent Internet user, do not buy this book. If you are a beginner or looking to get a home Internet connection, then this is the book for you. If you know people who could benefit from the Internet and you want to sell them on the Internet, get them to read *The Internet Companion*. Finally, *The Internet Companion* should reach many people due to its price, which is more in line with what many people can afford than many of the other Internet books.

CD Offer

Those of you who attended the 1991 Summer USENIX conference in Nashville heard Paul Lansky give a keynote on ways of reconstructing our fundamental views of music given the new capabilities offered us by the high-tech world. During that talk, he played several examples of his own work, many of them not yet available. They have recently been published on a CD which contains *Table's Clear* (the kitchen-gamelan piece), *Night Traffic*, *Quakerbridge* (the shopping mall piece), and two other pieces.

The CD is called *Homebrew* and is published by Bridge Records, BDC 9035. Tower Records should carry it but you may also order it directly from Bridge for \$15, which includes shipping and handling within the U.S.

Bridge Records
GPO Box 1864
New York, NY 10116
phone: 516-487-1662

[Paul Lansky would appreciate any feedback you might have – paul@silvertone.princeton.edu]

SEDMS IV

**Call for Participation:
Symposium on Experiences with Distributed
and Multiprocessor Systems IV (SEDMS IV)
San Diego, California
September 23-24, 1993**

Sponsored by: The USENIX Association
In cooperation with: ACM SIGARCH, SIGOPS, SIG-
SOFT (Pending), ACM SIGCOMM, and IEEE-CS
Technical Committees on Distributed Processing,
Operating Systems, Software Engineering, and
Design Automation

Goals

The goal of this symposium is to bring together individuals who have built, are building, or will soon build distributed and multiprocessor systems. SEDMS IV will provide a forum for individuals to exchange information on their experiences, both good and bad, including experiences with coding aids, languages, debugging and testing technology, reuse of existing software, and performance analysis. The presentations should emphasize the lessons learned from use of such systems and tools.

Extra-long breaks between sessions and work-in-progress presentations will be provided to facilitate a workshop-like atmosphere during parts of the symposium. We will also have discussion panels on submitted themes.

Submissions

Six copies of each submission or panel proposal should be sent to the program chair (address below) to arrive no later than April 27, 1993. Submissions of full papers are invited on any topics related to the theme of the symposium. The committee will give preferential consideration to submissions describing experiences with actual systems. Papers describing purely theoretical

work will not be accepted. Panel proposals should include a description of the relevance to the goals of the SEDMS, and the qualifications of the participants suggested.

Important Dates

Submissions due	April 27, 1993
Notifications mailed	June 14, 1993
Camera ready copy due	July 20, 1993

For further information, contact

David Cohn (Program Chair)
Computer Science and Engineering Dept.
University of Notre Dame
Notre Dame, IN 46556
(219) 239-6694
<dlc@cse.nd.edu>

Peter Reiher (General Chair)
Computer Science Dept.
Boelter Hall, UCLA
Los Angeles, CA 90024
(310) 206-8696
<reiher@wells.cs.ucla.edu>

Program Committee

John R. Nicol, *GTE Laboratories, Inc.*
Volker Tschammer, *GMD FOKUS Berlin*
Dag Johansen, *University of Tromso*
Karsten Schwan, *Georgia Tech*
Partha Dasgupta, *Arizona State University*
Brett Fleisch, *UC, Riverside*
David Pitts, *UMass Lowell*
Debra Hensgen, *University of Cincinnati*
John Barr, *Motorola*
Marc Pucci, *Bellcore*
Michael Scott, *University of Rochester*
Mike O'Dell, *Bell Communications Research*
Roy Campbell, *University of Illinois*
Ed Lazowska, *University of Washington*

Summer 1993 Conference

**Call for Papers:
USENIX Summer 1993
Technical Conference
Cincinnati, Ohio
June 21-25, 1993**

Evolving New User Interface Technologies For UNIX

A little over ten years ago UNIX encountered the bitmap display and the mouse. Developments since then, such as the X window system, didn't try to change UNIX. Rather they layered on it to cope with the demands of the new user interface technology. After ten years this doesn't appear to be a successful strategy. UNIX has industry-leading user interfaces and a horde of new user interface technologies are arriving.

Radical thinking and new operating system capabilities are needed to support new user interface technologies. Communicating with the user is a real-time problem, why aren't we using the emerging real-time capabilities of UNIX to support it? Are UNIX byte-string files adequate, or do we need a generalized file attribute model? Can users really navigate a file name space that is a rooted tree of all the files in the Internet?

As usual at the USENIX Conferences, we are interested in papers describing new and interesting developments in open operating systems. But in Cincinnati we're particularly interested in papers addressing the evolution of operating systems to support new and effective user interfaces.

Conference Program Committee

Program Chair:

David S. H. Rosenthal, *SunSoft Inc.*

Matthew Blaze, *AT&T Bell Laboratories*

Nathaniel Borenstein, *Bellcore*

Bob Gray, *U.S. West Advanced Technologies*

Steve Kleiman, *SunSoft Inc.*

John Kohl, *UC Berkeley*

Marshall Kirk McKusick, *UC Berkeley*

Jeffrey Mogul, *Digital Equipment Corporation*

J. R. Oldroyd, *Instruction Set*

Pat Parseghian, *AT&T Bell Laboratories*

Dennis Ritchie, *AT&T Bell Laboratories*

Important Dates

Dates For Refereed Paper Submissions

Extended Abstracts Due: February 2, 1993
Notification to Authors: February 27, 1993
Camera-ready Papers Due: April 14, 1993

How To Submit A Refereed Paper

Authors of papers to be presented at the technical sessions and published in the proceedings must submit by February 2, 1993 one copy of an extended abstract via at least two of the following methods:

E-mail: summer93papers@usenix.org

FAX: (510) 548-5738

Mail: Summer 93 USENIX

USENIX Association

2560 Ninth St, Suite 215

Berkeley, CA 94710 U.S.A.

The schedule for reviewing submissions for the conference is very short, and reviewers don't have time to read full papers. The object of an extended abstract is to convince the reviewers that a good paper and 25-minute presentation will result. They need to know that authors:

- are attacking a significant problem.
- are familiar with the current literature about the problem.
- have devised an original solution.
- have implemented it and, if appropriate, characterized its performance.
- have drawn appropriate conclusions about what they have learned and why it is important.

As at all USENIX conferences, papers that analyze problem areas and draw important conclusions from practical experience are welcome. Note that the USENIX conference, like most conferences and journals, considers it unethical to submit the same paper simultaneously to more than one conference or publication or to submit a paper that has been or will be published elsewhere.

The extended abstract must be 5 manuscript pages (single side) or less in length. Only the first 5 pages of your submission will be sent to the reviewers. The full paper may be attached to the extended abstract; it will not be sent to the reviewers but may be helpful during final evaluation.

The extended abstract should represent the paper in "short form." It should include the abstract as it will appear in the final paper. Supporting material may be in note form. Authors should include references to establish that they are familiar with the literature, and, if appropriate, performance data to establish that they have a working implementation and measurement tools.

Every submission should include one additional page containing:

- The name, surface mail address, daytime and evening telephone numbers, e-mail address and (if available) fax number of one of the authors, who will act as the contact to the program committee.
- An indication of which, if any, of the authors are full-time students
- A list of audio/visual equipment desired beyond a microphone and an overhead projector.

Authors of accepted submissions will be notified by February 27, 1993. They will promptly receive instructions for preparing camera-ready copy of an 8-12 page final paper, which must be received by April 14, 1993.

Inquiries about submissions to the USENIX Summer 1993 Conference may be made by e-mail to david@usenix.org or to (510) 528-8649. You may request a sample extended abstract by telephoning (510) 528-8649 or by fax to (510) 548-5738.

Invited Talks

Invited Talks Coordinators:
Tom Cargill, Consultant (303) 494-3239
Bob Gray, US WEST (303) 541-6014
E-mail to: ITusenix@usenix.org

As part of the technical sessions, a full series of invited talks provide introductory and advanced information about a variety of interesting topics, such as using standard UNIX tools, tackling system administration difficulties, or employing specialized applications. We welcome suggestions for topics as well as request proposals for particular Talks. In your proposal, state the main focus, include a brief outline, and be sure to emphasize why your topic is of general interest to our community.

For More Information

Materials containing all details of the technical and tutorial program, conference registration, hotel and airline discount and reservation information will be mailed at the end of March 1993. If you wish to receive the pre-registration materials, please contact the USENIX Conference Office

Summer 1993 Vendor Displays

The USENIX Vendor Display will provide a relaxed environment in which conference attendees and vendor technical support people have time to talk together and learn from one another. This is an exceptional opportunity for receiving feedback on new development from USENIX's technically astute conference attendees.

Only a small number of vendors will be able to participate. If your company would like to display its products and services, please contact:

Cynthia Deno
Tel: (408) 335-9445
FAX: (408) 335-2163
E-mail: cynthia@usenix.org

Mobile Computing Symposium

**Call for Papers:
Symposium on Mobile & Location-
Independent Computing
Cambridge, MA
August 2-3, 1993**

Much of the growth of UNIX has been due to its support for casual communications, thus fostering cooperative work within a location-independent framework. The latest incarnation of location independence is "Mobile Computing." Distributed computing, now fashionable in other circles, was pioneered by the UNIX community. Support for Mobile Computing is the next logical step in assuring the role of UNIX as the operating system that offers a rich and complete feature set.

Progress in Mobile Computing is everywhere evident, both in academic and non-academic circles. We intend to concentrate on it in a true state-of-the-art symposium and technical free-for-all on what it takes to make Mobile Computing work and work right. The workshop will address many issues and ongoing developments, including, but not limited to:

- Naming (e.g. Prospero or OSF/DCE DNS)
- Wide area information distribution (e.g. WAIS and archie)
- Security (e.g. authentication based on devices and digital signature services)
- User locatability (e.g. paging systems and active badges)
- Rendezvous (e.g. videoconferencing over the internet and various groupware efforts)
- Networking and Connectability (e.g. the new IETF routing work, movement of "sockets" from site to site, and the rumored advent of IP connections from airplanes)
- Portable tiny devices (e.g. the various palm-tops and personal information assistants)

As is usual for a USENIX symposium, we are looking for new and arresting developments in systems that directly contribute to a technical understanding of Mobile Computing. UNIX will be

the lingua franca of discussion, but we are eager for progress from other world views to be presented as well. This symposium will have limited attendance.

Extended abstracts of 1,500-2,500 words (9,000-15,000 bytes or 3-5 pages) should be sent to Dan Geer at the address below (those submitting hardcopy abstracts must send five copies). Shorter abstracts run a significant risk of rejection as there will be little on which the program committee can base an opinion.

Dates For Refereed Paper Submissions

April 19, 1993	Extended abstracts due
May 3, 1993	Notification to authors
June 14, 1993	Camera-ready copy due

Program Chair:

Dan Geer, Geer Zolot Associates

Vice-Program Chair:

Clement Cole, Locus Computing Corporation

Information:

For further information about the symposium, contact the Program Chair:

Daniel E. Geer
Geer Zolot Associates
200 Portland Street
Boston, MA 02114
Email: geer@world.std.com
Telephone: +1 617 367 2010
FAX: +1 617 367 6131

USENIX WINTER 1993 TECHNICAL CONFERENCE

THE
☆☆☆

CHALLENGE OF INNOVATION

SAN DIEGO • CALIFORNIA • JANUARY 25-29, 1993

☆☆☆ TUTORIALS ☆☆☆

MONDAY & TUESDAY, JANUARY 25 & 26, 1993

- Essential UNIX Programming
- OSF's Distributed Computing Environment (DCE)
Using, Managing, and Implementing NFS
OSF/1 Internals
- Programming with the X Window System
- Symmetric Multiprocessing and Caching in UNIX Kernels
- SVR4 Internals-The VFS and Process Subsystems
Topics in UNIX System Security
- Essentials of Practical Perl Programming
- Topics in Advanced System Administration
- UNIX Network Programming
- OSF's Distributed Management Environment (DME)
Distributed File System Administration with DCE/DFS
4.4BSD Kernel Internals
- Tcl and Tk: A New Approach to X11 and GUI Programming
Micro-Kernel Technology
- SVR4 Internals-The VM and I/O Subsystems
Network Security: The Kerberos Approach
- Introduction to Threads and Threads Programming
Managing the Domain Name System

☆☆☆ TECHNICAL SESSIONS ☆☆☆

WEDNESDAY, THURSDAY & FRIDAY, JANUARY 27, 28 & 29, 1993

☆☆☆ TECHNOLOGY ☆☆☆

- KEYNOTE: Pen-based Computing Robert Carr, *Go Corporation*
Hello World Internationalized
The Organization of Networks in Plan 9
An OSF/1 UNIX for MPP Systems
Improved Libraries for Dictionaries
The Nachos Instructional Operating System
An Object-Oriented UNIX Implementation
Invited Talk: Internationalization
- Invited Talk: Highlights from the USENIX MicroKernel Workshop
Invited Talk: Multimedia Mail: MIME & Metamail
Invited Talk: TCP/IP Networking
- Invited Talk: 1000 Year History of Graphics Languages
Invited Talk: A History of UNIX
Invited Talk: Object-Oriented Databases
- ☆☆☆ MOBILE COMPUTING TRENDS ☆☆☆
A Mobile Internetworking Architecture
PhoneStation-Moving the Telephone onto the Virtual Desktop
Mobile Computing Using Internet Packet Forwarding
- ☆☆☆ THREADS ☆☆☆
Pitfalls in Multithreading SVR4 STREAMS
Warlock-A Static Data Race Analysis Tool
A Library Implementation of POSIX Threads under UNIX

☆☆☆ FILESYSTEMS ☆☆☆

- Faster AFS
The AutoCacher: A File Cache for NFS
The Design and Implementation of the Inversion File System
Operating System Support for Portable Filesystem Extensions
An Implementation of a Log-Structured File System for UNIX
File Systems in User Space
HighLight: A Log-Structured File System for Tertiary Storage
Using Online Compression to Extend Physical Memory
Invited Talk: USENIX Filesystem Workshop Highlights

☆☆☆ GRAPHICS ☆☆☆

- Jgraph-A Filter for Plotting Graphs in PostScript
A Smart Frame Buffer
Wafe-An X Toolkit Based Frontend for Applications
Design and Implementation of a Multi-Threaded Xlib
Invited Talk: Visualization Software

☆☆☆ PROGRAMMING AND TESTING ☆☆☆

- Es: A Shell with Higher Order Functions
DUEL-A Very High Level Debugging Language
The San Diego "Zoo": A Multicomputer Test Suite
Linking Shared Segments
Glish: A Software Bus For Loosely Coupled Systems

☆☆☆ SYSTEM ADMINISTRATION ☆☆☆

- Security and Multilevel Storage and Communications
Handling Removable Media in Solaris
An Advanced Tape Cataloging System for UNIX Systems
Fremont: A System for Network Discovery
Essence: A Resource Discovery System
The Enterprise Distributed User Directory Service
Invited Talk: USENIX LISA VI Highlights
Invited Talk: The Odin System for makefiles
Invited Talk: Resource Discovery

☆☆☆ PERFORMANCE AND MONITORING ☆☆☆

- The BSD Packet Filter
UNIX Kernel Support for OLTP Performance
A Study of Overheads in DECStation Network Software
Exploiting In-Kernel Data Paths for Better Performance
Hardware Profiling of Kernels
A Randomized Sampling Clock For Code Profiling
Fault Interpretation: Fine-grain Monitoring of Page Access
UNIX Disk Access Patterns
Efficient Kernel Memory on Shared-Memory MPs
An Analysis of File Migration

☆☆☆ INTELLECTUAL PROPERTY ☆☆☆

- Presentation & Panel Discussion: Intellectual Property

USENIX, THE UNIX AND ADVANCED COMPUTING SYSTEMS PROFESSIONAL AND TECHNICAL ASSOCIATION

☆☆☆ FOR COMPLETE INFORMATION AND TO REGISTER, PLEASE CONTACT: ☆☆☆
USENIX CONFERENCE OFFICE, 22672 LAMBERT ST., SUITE 613, EL TORO, CA 92630 U.S.A.
TELEPHONE: (714) 588-8649; FAX: (714) 588-9706; EMAIL: conference@usenix.org
OFFICE HOURS: MONDAY - FRIDAY, 8:30 AM - 5:00 PM PACIFIC TIME

Computing Systems:

Call for Papers for Special Issues Call for Special Issue Proposals

Special Issue on Collaborative Computing Systems and Applications

A special issue of the journal *Computing Systems* to be published in 1993 will be devoted to "Collaborative Computing Systems and Applications." Papers on all aspects of design, implementation, and experiences with these systems are solicited for the issue. The deadline for submissions is December 22, 1992; because of the holiday, papers submitted after this deadline will not be considered. Prospective authors should send five copies of their papers to the guest editor:

Professor Prasun Dewan
1398 Computer Sciences Building
Department of Computer Sciences
Purdue University
W. Lafayette, IN 47907-1398
(317) 494-6014
<pd@cs.purdue.edu>

Submissions should not have appeared in other archival publications prior to their submission. Papers developed from earlier conference, symposia and workshop presentations are welcome.

Special Issue on Security and Integrity in Open Systems

A special issue of the journal *Computing Systems* to be published in 1993 will be devoted to "Security and Integrity of Open Systems." Papers on all aspects of policy, issues, theory, design, implementation, and experiences with security and integrity in open systems are solicited for the issue. The deadline for submissions is March 1, 1993; papers submitted after this deadline will not be considered. Prospective authors should send five copies of their papers to the guest editor:

Professor Matt Bishop
Mathematics and Computer Science
Dartmouth College
6188 Bradley Hall
Hanover, NH 03755-3551
(603) 646-3267
<Matt.Bishop@dartmouth.edu>

Submissions should not have appeared in other archival publications prior to their submission. Papers developed from earlier conference, symposia and workshop presentations are welcome.

Computing Systems:

Call for Special Issue Proposals

Call for Proposals for Special Issues

Gene Spafford, Associate Editor of *Computing Systems*, is seeking proposals for special issues of the journal for 1994. Proposals should propose a theme and discuss its relevance and importance to the readership of the journal. Each proposal should further specify potential sources of articles, and describe the qualifications of the proposer to act as guest editor. Questions and proposals should be directed to:

Professor Eugene Spafford
Department of Computer Sciences
1398 Computer Sciences Building
Purdue University
West Lafayette, IN 47907-1398
(317) 494-7825
<spaf@cs.purdue.edu>

About *Computing Systems*

Computing Systems (ISSN 0895-6340) is a refereed, quarterly journal published by the University of California Press for the USENIX Association. USENIX is a professional and technical association of individuals and institutions concerned with breeding innovation in the UNIX tradition.

Computing Systems is dedicated to the analysis and understanding of the theory, design, art, engineering and implementation of advanced computing systems, with an emphasis on systems inspired or influenced by the UNIX tradition. The journal's content includes coverage of topics in operating systems, architecture, networking, interfaces, programming languages, and sophisticated applications.

Now in its fifth year of publication, *Computing Systems* is regularly distributed to 4900 individual subscribers and over 600 institutional subscribers (libraries, research labs, etc.) around the world. Special topic issues are often distributed more widely.

The editor-in-chief of *Computing Systems* is Mike O'Dell of Bellcore. Gene Spafford of Purdue University is Associate Editor, and Peter Salus of the Sun User Group is the Managing Editor.

Editorial correspondence and subscription orders should be addressed to:

Computing Systems
2560 Ninth Street, Suite 215
Berkeley, CA 94710
510/528-8649
FAX 510/548-4738
office@usenix.org

Special Offer: Computing Systems

USENIX and the University of California Press are offering a substantial discount for a limited time on the 1988-1991 volumes (1-4) of *Computing Systems*, as follows:

- 50% off the complete 4 volume set = \$112
- 40% off single whole volumes (4 issues) = \$33.60 each
- 35% off single issues = \$9.10

To order back issues please reference the listing of articles on the adjacent page and fill out the form below:

I would like to order:	Amount Due
<input type="checkbox"/> Volumes 1-4, 1988-1991, @\$112 each	\$ _____
<input type="checkbox"/> Single volume, number _____ @\$33.60 each	
_____ @\$33.60	
_____ @\$33.60	
_____ @\$33.60	_____
<input type="checkbox"/> Single issue, vol. _____ num. _____ @\$9.10 each	
_____ @\$9.10	
_____ @\$9.10	
_____ @\$9.10	_____
Order Sub Total	_____
Add CA Sales Tax	_____
Additional International Postage** <input type="checkbox"/> Surface <input type="checkbox"/> Air	_____
Total amount enclosed \$	_____
Payment Options:	
_____ Check enclosed payable to <i>Computing Systems</i> .	
_____ Credit card: _____ Visa _____ MasterCard	
Account # _____	Expiration Date _____
Signature _____	
**International order? Please add one of the following amounts for postage:	
	Surface Air Freight
Complete 4 Volume Set:	\$23.50 \$65.00
Single Volume:	\$ 5.00 \$36.00
Single Issue:	\$ 1.25 \$ 9.00
Please make your payment in U.S. currency by: a check drawn on a U.S. bank, charge (Visa or MasterCard), or international postal money order.	
Ship To:	Name _____
	Address _____

	City _____ State/Country _____ Zip _____
Please send this form along with your payment to:	
USENIX Association	Fax: 510/548-5738
2560 Ninth Street, Suite 215	Phone: 510/528-8649
Berkeley, CA 94710	email: office@usenix.org

VOLUME 1

Number 1, Winter 1988: "The Synthesis Kernel", Pu, Massalin, Ioannidis
"Language and Operating System Features for Real-time Programming", Donner, Jameson
"Dynamics for Computer Graphics: A Tutorial", Wilhelms

Number 2, Spring 1988: "Enhanced Resource Sharing in UNIX", Barton, Wagner
"Design and Implementation of Parallel Make", Baalbergen
"Yacc Meets C++", Johnson
"Watchdogs - Extending the UNIX File System", Bershad, Pinkerton
"Controversy: Can UNIX survive secret source code?", Lesk

Number 3, Summer, 1988: "GRAB - Inverted Indexes with Low Storage Overhead", Lesk
"An Application of a Fast Data Encryption Standard Implementation", Bishop
"Effects of a copy-on-write Memory Management on the Response Time of UNIX fork Operations", Smith, Maguire
"Controversy: Window Systems Should be Transparent", Pike

Number 4, Fall 1988: "CHORUS Distributed Operating Systems", Rozier, et al.
"Type-safe Linkage for C++", Stroustrup
"An Unorthodox Approach to Undergraduate Software Engineering Instruction", Morris

VOLUME 2

Number 1, Winter 1989: "Developing Applications for Heterogeneous Machine Networks: The Durra Environment", Barbacci, et al.
"A Hypertext System for UNIX", Brown
"Parameterized Types for C++", Stroustrup

Number 2, Spring 1989: "Page Makeup by Postprocessing Text Formatter Output", Kernighan, Van Wyk
"A Concurrent Window System", Pike
"Experience with Viruses on UNIX Systems", Duff
"Virology 101", McIlroy

Number 3, Summer 1989: "The Evolution of C++: 1985 to 1989", Stroustrup
"Heuristics for Disk Drive Positioning in 4.3BSD", Stevens

Number 4, Fall 1989: "SOS: An Object-Oriented Operating System - Assessment and Perspectives", Shapiro, et al.
"Data Structures in the Icon Programming Language", Griswold
"Multiple Inheritance for C++", Stroustrup

VOLUME 3

Number 1, Winter 1990: "The Design and Implementation of the Clouds Distributed Operating System", Dasgupta, et al.
"Using Hints in DUNE Remote Procedure Calls", Pucci, Alberi
"Mach/4.3BSD: A Conservative Approach to Parallelization", Boykin, Langerman
"Implementation Issues for the Psyche Multiprocessor Operating System", Scott, et al.
"Fine-Grain Adaptive Scheduling using Feedback", Massalin, Pu

Number 2, Spring 1990: "Little Languages for Music", Langston
"The Personal Orchestra, or Audio Data Compression by 10,000:1", Hawley
"Keynote - A Language and Extensible Graphic Editor for Music", Thompson
"Controversy: Portability - A No Longer Solved Problem", Feldman, Gentleman

Number 3, Summer 1990: "Process Synchronization in the UTS Kernel", Ruane
"A Concurrent Programming Support for Distributed Systems", Spezzano, Talia, Vanneschi
"Distributed Spooling in a Heterogeneous Environment", Wagner

Number 4, Fall 1990: "An Experimental Implementation of the Tilde Naming System", Comer, Droms, Murtagh
"An Object Model for Conventional Operating Systems", Dewan, Vasilik
"A Comparison of Basic CPU Scheduling Algorithms for Multiprocessor UNIX", Curran, Stumm

VOLUME 4

Number 1, Winter 1991: "A System for Algorithm Animation", Bentley, Kernighan
"Architecture and Implementation of Guide, and Object-Oriented Distributed System", Balter, et al.
"Controversy: The Case Against Multiple Inheritance in C++", Cargill

Number 2, Spring 1991: "expect: Scripts for Controlling Interactive Processes", Libes
"An ASCII Database for Fast Queries of Relatively Stable Data", Herrin, Finkel
"Controversy: The Case for Multiple Inheritance in C++", Waldo

Number 3, Summer 1991: "Experience Developing the RP3 Operating System", Byrant, Chang, Rosenberg
"Configurable Data Manipulation in an Attached Multiprocessor", Pucci
"Distributed Programming with Objects and Threads in the Clouds System", Dasgupta, et al.
"Evolution of a Communication System for Distributed Transaction Processing in Raid", Bhargava, Zhang, Mafla
"Measured Performance of Caching in the Sprite Network File System", Welch

Number 4, Fall 1991: "A Comparison of Two Distributed Systems: Amoeba and Sprite", Douglass, et al.
"The Software Design Laboratory", Smith
"Swift: Using Distributed Disk Striping to Provide High I/O Data Rates", Cabrera, Long

Publications Order Form

CONFERENCE & WORKSHOP PROCEEDINGS

Qty	Proceedings		Member Price	Non-Member Price	Subtotal	Overseas Postage	Total
WINTERS/SUMMER CONFERENCES							
___	San Antonio	Summer '92	\$23	\$30	\$ _____	\$14	\$ _____
___	San Francisco	Winter '92	30	39	\$ _____	22	\$ _____
___	Nashville	Summer '91	32	38	\$ _____	22	\$ _____
___	Dallas	Winter '91	28	34	\$ _____	18	\$ _____
___	Anaheim	Summer '90	22	22	\$ _____	15	\$ _____
___	Washington, DC	Winter '90	25	25	\$ _____	15	\$ _____
___	Baltimore	Summer '89	20	20	\$ _____	15	\$ _____
___	San Diego	Winter '89	30	30	\$ _____	20	\$ _____
___	San Francisco	Summer '88	29	29	\$ _____	20	\$ _____
___	Dallas	Winter '88	26	26	\$ _____	15	\$ _____
___	Phoenix	Summer '87	35	35	\$ _____	20	\$ _____
___	Washington, DC	Winter '87	10	10	\$ _____	15	\$ _____
___	Atlanta	Summer '86	37	37	\$ _____	20	\$ _____
___	Denver	Winter '86	25	25	\$ _____	15	\$ _____
___	Portland	Summer '85	45	45	\$ _____	25	\$ _____
___	Dallas	Winter '85	15	15	\$ _____	10	\$ _____
___	Salt Lake City	Summer '84	29	29	\$ _____	20	\$ _____
___	Washington, DC	Winter '84	25	25	\$ _____	15	\$ _____
___	Toronto	Summer '83	32	32	\$ _____	20	\$ _____
___	San Diego	Winter '83	28	28	\$ _____	15	\$ _____
LARGE INSTALLATION SYSTEMS ADMINISTRATION							
___	LISA VI	Oct. '92	23	30	\$ _____	12	\$ _____
___	LISA V	Sept. '91	20	23	\$ _____	11	\$ _____
___	LISA IV	Oct. '90	15	18	\$ _____	8	\$ _____
___	LISA III	Sept. '89	13	13	\$ _____	9	\$ _____
___	LISA II	Nov. '88	8	8	\$ _____	5	\$ _____
___	LISA I	April '87	4	4	\$ _____	5	\$ _____
C++							
___	C++ Conference	Aug. '92	30	39	\$ _____	20	\$ _____
___	C++ Conference	Apr. '91	22	26	\$ _____	11	\$ _____
___	C++ Conference	Apr. '90	28	28	\$ _____	18	\$ _____
___	C++ Conference	Oct. '88	30	30	\$ _____	20	\$ _____
___	C++ Workshop	Nov. '87	30	30	\$ _____	20	\$ _____
SECURITY							
___	UNIX Security III	Sept. '92	30	39	\$ _____	11	\$ _____
___	UNIX Security II	Aug. '90	13	16	\$ _____	8	\$ _____
___	UNIX Security	Aug. '88	7	7	\$ _____	5	\$ _____
MACH							
___	Mach Symposium	Nov. '91	24	28	\$ _____	14	\$ _____
___	Mach Workshop	Oct. '90	17	20	\$ _____	9	\$ _____

Qty	Proceedings		Member Price	Non-Member*	Subtotal	Overseas Postage	Total
DISTRIBUTED & MULTIPROCESSOR SYSTEMS (SEDMS)							
___	SEDMS III	Mar. '92	30	36	\$ _____	20	\$ _____
___	SEDMS II	Mar. '91	30	36	\$ _____	20	\$ _____
___	SEDMS	Oct. '89	30	30	\$ _____	20	\$ _____
GRAPHICS							
___	Graphics Workshop V	Nov. '89	18	18	\$ _____	10	\$ _____
___	Graphics IV	Oct. '87	10	10	\$ _____	10	\$ _____
___	Graphics III	Nov. '86	10	10	\$ _____	5	\$ _____
___	Graphics II	Dec. '85	7	7	\$ _____	5	\$ _____
OTHER WORKSHOPS							
___	File Systems	May '92	15	20	\$ _____	9	\$ _____
___	Micro-Kernel & Other Kernel Arch.	April '92	30	39	\$ _____	20	\$ _____
___	UNIX Transaction Processing	May '89	12	12	\$ _____	8	\$ _____
___	Software Management	Apr. '89	20	20	\$ _____	15	\$ _____
___	UNIX & Supercomputers	Sept. '88	20	20	\$ _____	10	\$ _____



Discounts are available for bulk orders. Please inquire.

Total price of Proceedings _____ **
Calif. residents add sales tax _____
Total overseas postage _____
Total enclosed _____

**If you are paying member price, please include member's name and/or membership number _____

PAYMENT OPTIONS*

___ Check enclosed- payable to USENIX Association

___ Charge my: ___ VISA  ___ MC  Account # _____ Exp.Date _____

___ Purchase order enclosed Signature _____

- * Outside the USA? Please make your payment in US currency by one of the following:
- Check - issued by a local branch of a US Bank
 - Charge (VISA, MasterCard, or foreign equivalent)
 - International postal money order

Shipping Information

Ship to:

Please allow 2-3 weeks for delivery. Overseas orders are shipped via air printed matter.

Please mail or fax this order form with payment to:

USENIX Association
 2560 Ninth Street, Suite 215
 Berkeley, CA 94710
 FAX 510/548-5738

- If you are not a member and wish to receive our membership information packet, please check this box.

Membership Application

MEMBERSHIP INFORMATION

Any individual or institution may become a member by filling out an application form and paying the appropriate annual fee.

There are five classes of membership:

Student: \$20
Open to any full-time student at an accredited educational institution. A copy of the current student I.D. card must be provided.

Individual: \$65
Open to any individual or institution. Individual Members may vote.

Corporate: \$325
Corporate Membership is open to any individual or institution.

Educational: \$160
Educational Membership is open to accredited educational institutions.

Supporting: \$1000
Open to any individual or institution that wants to support the Association to a greater degree than through the Corporate Membership fee.

Corporate, Educational and Supporting members receive all services available to Individual Members, plus copies of the proceedings from all conferences and workshops that are held during the term of membership.

MEMBERSHIP APPLICATION

New Renewal

Name _____

Address _____

City _____ State _____ Zip _____ Country _____

Phone _____ email address: _____

- | | |
|---|--|
| <input type="checkbox"/> \$20 Student (full-time)
(with copy of I.D. card) | <input type="checkbox"/> \$160 Educational Institution |
| <input type="checkbox"/> \$65 Individual | <input type="checkbox"/> \$325 Corporate |
| | <input type="checkbox"/> \$1000 Supporting |

PAYMENT OPTIONS

- Check enclosed payable to USENIX Association. Purchase order enclosed (Educational and Corporate members only).

Please charge my: Visa MasterCard



Account # _____ Exp. Date _____

Signature _____

Outside the U.S.A.? Please make your payment in U.S. currency by one of the following:

- * Charge (Visa, MasterCard, or foreign equivalent)
- * International postal money order
- * Check - issued by a local branch of a U.S. Bank

- Please send me information on purchasing USENIX Software Distribution Tapes. 10/92
 Please send me information on purchasing the Second Berkeley Software Distribution Tape (version 2.11).

USENIX Mailing List

- I do not want my address made available to other members.
 I do not want my address made available for commercial mailings.

USENIX Online Library and Index

What Is It

The USENIX online index is an electronically available list of papers published by the USENIX Association and related groups. The index is kept as a simple ASCII file, in refer/bib format, sorted by author, and contains information about papers published in USENIX conference and workshop proceedings, newsletters, journal, and the like.

The index is updated approximately monthly.

How to Get the Index

The index is available online from UUNET, either via a mail server or anonymous ftp. The index is about 200KB, and available only in its entirety.

To get it as mail, send mail to *library@uunet.uu.net* with "send bibliography" as the contents of your message.

To get it via ftp from *ftp.uu.net*, login as "anonymous" with your email address as the password. Then:

```
ftp> cd library
250 CWD command successful.
ftp> get bibliography
```

This help file can be retrieved with "send help" or as the ftp file "help.bibliography".

(There is no person associated with the library address and it will never be read by human eyes.)

How to Access Information

To build the indices so you can easily access information, run "indxib" on the bibliography: *indxib file.name*. You can then pull information from the file by running "lookbib". You can either build refer files or run lookbib interactively.

For example, the following command would put all entries which refer to Smith into a file called "stuff":

```
echo smith | lookbib bibliography >
stuff
```

Or you could interact with the index by saying:
lookbib bibliography

It will ask you if you want instructions when it starts, answer yes. Then at the prompt, for example:

```
> smith
```

will list references to smith (upper/lower case doesn't matter).

To Get an Online Paper

As of this date, we have not yet set up the online papers. When this capability is provided, we will announce it on the net and these instructions will be updated with retrieval information.

Publications Indexed

USENIX: Conference proceedings, workshop and symposia proceedings, *Computing Systems* journal, newsletter

EurOpen (formerly EUUG – European Unix Users Group): Conference proceedings, newsletter (1982-1989)

Other sources are being continually evaluated and will be included as deemed suitable.

Fields Used in the Index

The standard bib/refer formats are used. These include:

A	Author (may be multiple entries)
T	Title of article
P	Page number(s)
W	Primary author's institution
I	Issuer/publisher
B	Conference proceedings or book title
J	Name of newsletter or journal
D	Date of publication or conference
C	Location of conference
V	Volume number
N	Number within volume
O	Other comments (e.g., "Abstract only")

These fields may be extended to include other information such as identifier for retrieval, keywords, online format of paper (PostScript, troff, etc.), language (if other than English), etc.

More Information

For additional information about the online index and library, and/or instructions for donating papers, contact: *index@usenix.org*

Or write to:
USENIX Association
2560 Ninth St., Suite 215
Berkeley CA 94710

Local User Groups

The Association will support local user groups by doing a mailing to assist in the formation of a new group and publishing information on local groups in ;login:. At least one member of the group must be a current member of the Association. Send additions and corrections to: login@usenix.org.

CA - Fresno:

The Central California UNIX Users Group consists of a uucp-based electronic mailing list to which members may post questions or information. For connection information:

Educational and governmental institutions:
Brent Auernheimer (209) 278-2573
brent@CSUFresno.edu or *csufres!brent*

Commerical institutions or individuals:
Gordon Crumal (209) 251-2648
csufres!gordon

CA - Orange County:

Meets the 2nd Monday of each month

UNIX Users Association of Southern California
Paul Muldoon (714) 556-1220 ext. 137
New Horizons Computer Learning Center
1231 E. Dyer Rd., Suite 140
Santa Ana, CA 92705

CO - Boulder:

Meets monthly at different sites. For meeting schedule, send email to fruug-info@fruug.org.

Front Range UNIX Users Group
Software Design & Analysis, Inc.
1113 Spruce St., Ste. 500
Boulder, CO 80302
Steve Gaede (303) 444-9100
gaede@fruug.org

D.C. - Washington, D.C.:

Meets 1st Tuesday of each month.

Washington Area UNIX Users Group
9811 Mallard Drive
Laurel, MD 20708
Alan Fedder (301) 953-3626

FL - Coral Springs

S. Shaw McQuinn (305) 344-8686

FL - Western:

Meets 1st Thursday of each month.

Florida West Coast UNIX Users Group
Richard Martino (813) 536-1776
Tony Becker (813) 799-1836
mcrsys!tony
Ed Gallizzi, Ph.D. (813) 864-8272
e.gallizzi@compmail.com
Jay Ts (813) 979-9169
uunet!pdn!tscs!metran!jan
Dave Lewis (407) 242-4372
dhl@ccd.harris.com

FL - Orlando:

Meets the 3rd Thursday of each month.

Central Florida UNIX Users Group
Mikel Manitus (407) 444-8448
mike@aaa.com

FL - Melbourne

Meets the 3rd Monday of every month.

Space Coast UNIX User's Group
Steve Lindsey (407) 242-4766
lindsey@vnet.ibm.com

KS or MO - Kansas:

Meets on 2nd Monday of each month.

Kansas City UNIX Users Group (KUUG)
813B Street
Blue Springs, MO 64015
(816) 235-5212
mlg@cstp.umkc.edu

GA - Atlanta:

Meets on the 1st Monday of each month in White Hall, Emory University.

Atlanta UNIX Users Group
P.O. Box 12241
Atlanta, GA 30355-2241
Mark Landry (404) 365-8108

MI - Detroit/Ann Arbor

Meets on the 2nd Thursday of each month in Ann Arbor.

Southeastern Michigan Sun Local Users Group
and Nameless UNIX Users Group
Steve Simmons office: (313)769-4086
home: (313) 426-8981
scs@lokkur.dexter.mi.us

MN - Minneapolis/St. Paul:

Meets the 1st Wednesday of each month.

UNIX Users of Minnesota
17130 Jordan Court
Lakeville, MN 55044
Robert A. Monio (612) 220-2427
pnessutt@dmshq.mn.org

MO - St. Louis:

St. Louis UNIX Users Group
P.O. Box 2182
St. Louis, MO 63158
Terry Linhardt (314) 772-4762
uunet!jgalst!terry

NE - Omaha:

Meets monthly.

/usr/group/nebraska
P.O. Box 31012
Omaha, NE 68132
Phillip Allendorfer (402) 423-1400

New England - Northern:

Meets monthly at different sites.

Peter Schmitt 603) 646-2085
Kiewit Computation Center
Dartmouth College
Hanover, HN 03755
Peter.Schmitt@dartvax!dartmouth.edu

NJ - Princeton:

Meets monthly.

Princeton UNIX Users Group
Mercer County Community College
1200 Old Trenton Road
Trenton, NJ 08690
Peter J. Holsberg (609) 586-4800
mccc!pjh

NM - Albuquerque:

ASIGUNIX meets every 3rd Wednesday
of each month.

Phil Hertz 505/275-0466.

NY - New York City:

Meets every other month in Manhattan.

Unigroup of New York City
G.P.O. Box 1931
New York, NY 10116

OK - Tulsa:

Meets 2nd Wednesday of each month.

Tulsa UNIX Users Group, \$USR
Stan Mason (918) 560-5329
tulsix!smason@drd.com
Mark Lawrence (918) 743-3013
mark@drd.com

TX - Austin:

Meets 3rd Thursday of each month.

Capital Area Central Texas UNIX Society
P.O. Box 9786
Austin, TX 78766-9786
officers@cactus.org
Tom Painter (512) 835-5457
president@cactus.org

TX - Dallas/Fort Worth:

Dallas/Fort Worth UNIX Users Group
660 Preston Forest, Suite 177
Dallas, TX 75230
Kevin Coyle (214) 991-5512
kevincd@shared.com

TX - Houston:

Meets 3rd Tuesday of each month.

Houston UNIX Users Group
(Hounix) answering machine (713) 684-6590
Bob Marcum, President (713) 270-8124
Chuck Bentley, Vice-president
(713) 789-8928 chuckb@hounix.uucp

WA - Seattle:

Meets monthly.

Seattle UNIX Group Membership Info.
Bill Campbell (206) 947-5591
6641 East Mercer
Mercer Island, WA 98040-0820
bill@celestial.com

CANADA - Toronto:

143 Baronwood Court
Brampton, Ont. Canada L6V 3H8
Evan Leibovitch (416) 452-0504
evan@telly.on.ca

LISA Groups

Back Bay LISA

Forum covering all aspects of System and Network Administration, for large and small installations. Meets Monthly, various locations in Boston.

JR. Oldroyd
The Instruction Set
601 Trapelo Road
Waltham MA 01254
(617) 890 4930
jr@inset.com

Mailing list: bblisa@inset.com
List Requests: bblisa-request@inset.com

BAY LISA

The Bay-LISA group meets monthly in Santa Clara, CA, to discuss topics of interest for administration of sites with more than 100 users and/or computers.

December 17: Paul Morarity: SOCKS
January 21: Best of LISA VI

Send e-mail to baylisa-info@sysadmin.com,
or you may contact:

Bjorn Satdeva
(408) 241-3111
bjorn@sysadmin.com

Statement of Ownership Management and Circulation, 10/1/92

Title: *login:*, Pub. No. 008334. Frequency: Bimonthly. Six issues published annually. Subscription price: \$50 individuals and institutions. Location of office of publication: 2560 Ninth Street, Suite 215, Berkeley, Alameda County, CA, 94710. Headquarters of publishers: Same. Publisher: USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710. Editor: Ellie Young, 2560 Ninth Street, Suite 215, Berkeley, CA 94710. Owner: USENIX Association. The purpose and function, and nonprofit status of the organization and the exempt status for Federal income tax purposes have not changed during the preceding 12 months.

Extent and nature of circulation	Average no. copies each issue preceding 12 mos.	Actual no. copies of single issue published nearest to filing date
A. Total no. copies	5655	5700
B. Paid circulation, Mail subs.	5208	5436
C. Total paid circulation	5208	5436
D. Free distribution	345	190
E. Total distribution	5553	5626
F. Copies not distributed	102	74
G. Total	5655	5700

I certify that the statements made by me above are correct and complete.
Ellie Young, Executive Director

Calendar of Events

1993

- Jan 11-15 IEEE 1003, New Orleans, LA
25-29 * USENIX, San Diego, CA
Feb 22-24 Sun Open Sys. Expo, Chicago, IL
Mar 8 -12 Interop, Washington, D.C.
15-19 UniForum, San Francisco, CA
29-
Apr 1 * UNIX Applications Development
Toronto, Canada
19-21 * Mach III, Santa Fe, NM
19-23 IEEE 1003
May 3 - 7 EurOpen, Seville, Spain
20-22 UniForum NZ, New Zealand
Jun 5-11 DECUS, Atlanta, GA
21-25 * USENIX, Cincinnati, OH
Jul 12-16 IEEE 1003
Aug 1 ACM Siggraph, Anaheim, CA
2 - 3 * Mobile & Location Independent
Computing, Cambridge, MA
23-27 Interop, San Francisco, CA
INET '93, San Francisco, CA
Sept 20-22 *Micro-Kernels II, San Diego, CA
23-24 * SEDMS IV, San Diego, CA
Oct 4-6 * UNIX Security Symposium IV
18-22 IEEE 1003
Nov 1- 5 * LISA VII
Autumn EurOpen/UniForum
Utrecht, Netherlands
Dec 4-10 DECUS, San Francisco, CA

1994

- Jan 17-21 * USENIX, San Francisco, CA
Mar 23-25 UniForum, San Francisco, CA
Apr 18-22 EurOpen
May 7-13 DECUS, New Orleans, LA
Jun 6-10 * USENIX, Boston, MA
Sep 12-16 Interop, San Francisco, CA
Autumn EurOpen/UniForum
Utrecht, Netherlands
Nov 12-18 DECUS, Anaheim, CA

1995

- Jan 16-20 * USENIX, New Orleans, LA
Feb 21-23 UniForum, Dallas, TX
May 1- 5 EurOpen
13-19 DECUS, New Orleans, LA
Jun 19-22 * USENIX, San Francisco, CA
Nov 2- 8 DECUS, San Francisco, CA

1996

- Jan 22-26 * USENIX, San Diego, CA
Mar 12-14 UniForum, San Francisco, CA
May 18-24 DECUS, Orlando, FL
Nov 16-22 DECUS, Anaheim, CA

This is a combined calendar of planned conferences, workshops, and standards meetings related to the UNIX operating system. If you have a UNIX-related event that you wish to publicize, please contact login@usenix.org. Please provide your information in the same format as above. This calendar has been compiled with the assistance of Alain Williams of EurOpen.

* = events sponsored by the USENIX Association.

ACE: Advanced Computing Environments
ACM: Association for Computing Machinery
AFUU: Association Francaise des Utilisateurs d'UNIX
AUUG: Australian UNIX Users Group

DECUS: Digital Equipment Computer Users Society
EurOpen: European Forum for Open Systems
GUUG: German UNIX Systems User Group
IEEE: Institute of Electrical and Electronics Engineers

IETF: Internet Engineering Task Force
INET: Internet Society
Interex: Intl Assoc.-Hewlett-Packard Comp.Users
JUS: Japan UNIX Society

LISA: USENIX Systems Administration Conference
SEDMS: Symposium on Experiences with Distributed
and Multiprocessor Systems
UKUUG: United Kingdom UNIX Systems Users Group
UniForum: International Association of UNIX and
Open Systems Professionals

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

**Second Class Postage
PAID
At Berkeley, California
and Additional Offices**

POSTMASTER: Send address changes to ;login:, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710

What's Inside?

C++ Conference Report

Summer 1993 Conference Program

Standards Activity Update